SFA Modernization Partner

United States Department of Education

Student Financial Assistance

Integrated Technical Architecture

Build and Test Report

Task Order #16

Deliverable # 16.1.7

November 21, 2000

Table of Contents

List of Figures

List of Tables

1 Introduction

The Build and Test phase of Release 1 of the Integrated Technical Architecture (ITA) validates the architecture design, services, and interfaces provided by the ITA to support the Modernization effort of the Student Financial Assistance (SFA) Information Technology (IT) Enterprise.

1.1. Objectives

The objective of this Build and Test Report (BTR) deliverable is to provide the information necessary to execute and validate the ITA Build and Test Phase, which shall demonstrate that the ITA architecture provides the functional capabilities as defined in the ITA Detailed Design Document (Deliverable 16.1.2).

During the Build and Test phase all Release 1 architectural components shall be integrated into the ITA and shall be verified to execute properly.

The tests outlined in this report are based on the functional scenarios that were performed within the ITA to support the operational capabilities of the enterprise required for Release 1. Some of the testes defined in this report will be executed Post Release 1 since the functionality provided has not been fully implemented, nor is it required to support the Release 1 applications.

1.2. Scope

The Build portion will ensure that all required components defined in the ITA are installed, configured and operational. The Release 1 Build is documented in Deliverable 16.1.6 - ITA Software Installation / Configuration Report.

The Test portion will ensure that the actual outputs produced conform to the expected outputs as defined by the individual test scenarios, and that the system performs to a required standard under a variety of conditions as defined by the individual system process test specifications.

1.3. Approach

The following approach was used to develop the ITA Build and Test Report:

- Review of the ITA architecture from a services and interfaces perspective.
- Review of the functional services defined in each architecture domain. Based on these services the Build and Test team identified the interface definitions between products.
- Development of test scenarios that execute the touch points between products.

1.4. Organization of this Document

This remainder of this document is divided into three parts.

- ITA Test Methodology
- Test scenarios for Release 1
- Test scenarios for Post Release 1

2 ITA Test Methodology

2.1. Testing Process

The Build and Test procedures will focus on validating the ITA architectural design. For each identified Release 1 test scenario the following sections will be defined:

- Test Scenario Description
- Test Scenario Dependencies and Required Resources
- Test Scenario Detailed Design Description
- Test Scenario Inputs
- Test Scenario Expected Outputs
- Test Scenario Acceptance Criteria

2.1.1. Test Scenario Description

This section defines the objective(s) and a short textual description provides a high level view of the test to be performed, functions / services exercised, and any other pertinent aspects of the test scenario.

2.1.2. Test Scenario Dependencies and Required Resources

This section defines the dependencies that must be met prior to test execution, and defines the resources required in order to execute the test scenario.

2.1.3. Test Scenario Detailed Design Description

This section provides test scenario design detail via diagrams and text descriptions of what the diagrams represent. Each detail diagram depicts the test scenario's process flow by identifying each component and the interfaces involved. The text portion provides a description of what is occurring within each process module, what information is being shared, and how it is being transferred between products.

2.1.4. Test Scenario Inputs

This section describes the data required to execute the test scenario.

Test Procedure Steps

This section lists the individual detailed steps required in order to execute the scenario in a repeatable manner. Each step is divided into user action and expected observable result.

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 1 – Test Scenario # Sample – Procedure Steps

#	User Action	Expected Observable Result
1.	{ Description of the actual keystrokes, mouseclicks, or other action the user must execute } Examples …	{ Description of the expected observable result of the user's action. } Examples …
2.	Click in the USERNAME field Or Use the [TAB] key to move to the USERNAME field.	The cursor appears in the field.
3.	Type: *username*	Text is displayed in the field as input.
4.	Start the tail process on the idea log: **tail –f /home/***user***/app/idea.log**	Text is displayed on the screen – if the log is updated while being viewed screen contents will change.
5.	Watch the display of the log contents watching for xyz …	The text "This is the idea log for user: *user*" will be displayed.

2.1.5. Test Scenario Expected Results

This section includes any data to be displayed, reports or any other pertinent information which will be used to validate the results of the test scenario and should be related back to the objectives listed in the Test Description Section.

2.1.6. Test Scenario Acceptance Criteria

This section describes standards or measures against which the results can be validated and which demonstrate that the test results are correct and the functionality / services tested are consistent with the design of the architecture. These should be related back to the objectives listed in Test Description Section.

2.2. Test Environment Architecture Diagram

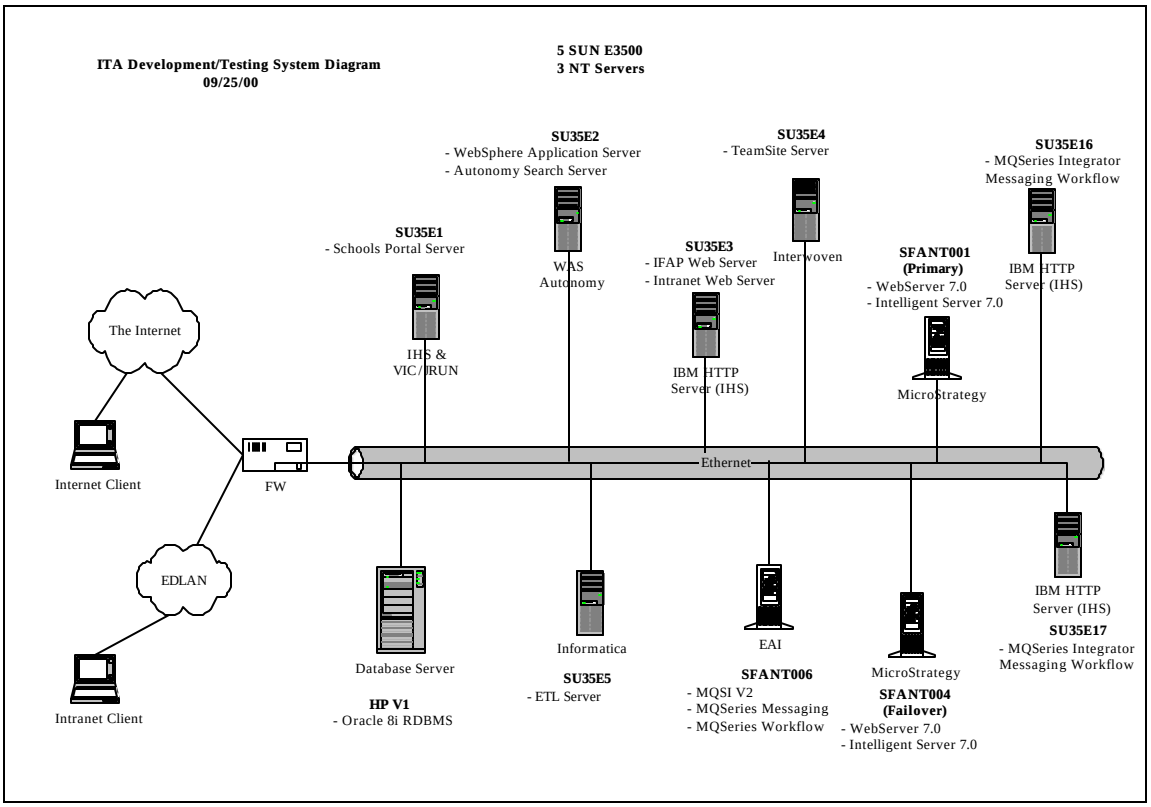

Figure 1 – Test Environment Architectural Diagram

2.3. Test Environment Software Versions

Table 2 – Test Environment Software Versions – Sun Solaris

Machine	IP Address	Logical Name	Installed Products
SU35E1	4.20.14.131	Portal Server	IHS, Viador, JRun, OpenDeploy Server, JDK, Servlet API
SU35E2	4.20.14.132	Application Server	Autonomy, WebSphere Advanced Edition, JDK, Servlet API, OpenDeploy Server
SU35E3	4.20.14.133	Web Server	IHS, OpenDeploy Server
SU35E4	4.20.14.134	Content Management	TeamSite, OpenDeploy Client, DataDeploy, Apache Web Server
SU35E5	4.20.14.135	Informatica Server	Informatica PowerCenter e

Table 3 – Test Environment Software Versions – HP-UX

Machine	IP Address	Logical Name	Installed Products
HPV1	4.20.14.59	Development Database Server	Oracle 8i SIDs installed -VICDEV, DLMDEV, DLMTST Targeted SIDs: VICTST, VICSTG, IFAPD, IFAPT, and IFAPS

Table 4 – Test Environment Software Versions – Windows NT

Machine	IP Address	Logical Name	Installed Products
SFANT001	4.20.14.244	MicroStrategy (Primary)	MicroStrategy Web Server, MicroStrategy Intelligence Server, MS IIS
SFANT004	4.20.14.248	MicroStrategy (Fail over)	MicroStrategy Web Server, MicroStrategy Intelligence Server, MS IIS
SFANT006	4.20.14.249	EAI Development Server	MQ Series Messaging, MQ Series Integrator, UDB Database

3 ITA Test Scenarios – Release 1

Below is a list of the Release 1 test scenarios that will be discussed. The sections that follow contain the information as outlined in Section 2.1.

Scenario # 1 – MicroStrategy – Access to Informatica Metadata

Scenario # 2 – MicroStrategy – OLAP Analysis over the Web

Scenario # 3 – MicroStrategy – Server Fail Over

Scenario # 4 – Interwoven – Push Content to TeamSite Staging Repository (Internal Test)

Scenario # 5 – Interwoven – Push Content to Production System (External Test)

Scenario # 6 – Viador – Portal Authentication

Scenario # 7 – Viador – Portal Socket Search

Scenario # 8 – WebSphere – Application Server JDBC Request

Scenario # 9 – WebSphere – Application Server Search

Scenario # 10 – WebSphere – Error Handling Report to User

Scenario # 11 – IHS – Static Content Flow

Scenario # 12 – IHS – Dynamic Content Flows

Scenario # 13 – Autonomy – Internal Content Retrieval (File System)

Scenario # 14 – Autonomy – DRE Query Validation (Local Host)

Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration)

Scenario # 16 – Autonomy – External Content Retrieval (Internet)

Scenario # 17 – Autonomy – External Content Scheduled Retrieval (Internet)

Autonomy – Fail Over Testing Report

3.1. Scenario # 1 – MicroStrategy – Access to Informatica Metadata

3.1.1. Test Scenario Description

This test will demonstrate that a user can view information from the Informatica metadata repository for attributes and facts in a MicroStrategy project. For a given attribute (an item in the data warehouse), a user can run a report (over the web) to view the extraction, transformation, and load information for that specific attribute.

The purpose of the test is to verify the ability to perform the metadata lookup services provided by the data warehouse architecture. The following components will be used in the performance of this test:

- MicroStrategy Web
- MicroStrategy Intelligence Server
- Microsoft Internet Information Server (IIS)
- Informatica Metadata Repository
- Oracle Database

3.1.2. Test Scenario Dependencies and Required Resources

Table 5 – Test Scenario # 1 – MicroStrategy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		A connection to the NT Server which holds MicroStrategy Web must be available from the user location (SFA LAN)
Personnel		✓	User
Test Data	✓		Informatica metadata repository must be populated with the ETL information for the CDS data in the data warehouse
Prior Testing	✓	✓	Validation of the configuration and operation of the eNetwork Dispatcher cluster for MicroStrategy
Hardware		✓	See Deliverable 16.1.2 Volume 4 - Data Warehouse Architecture
Software		✓	See Deliverable 16.1.2 Volume 4 - Data Warehouse Architecture
Time to be Allotted			5 minutes

3.1.3. Test Scenario Detailed Design Description

Using a web browser, a user will connect to the MicroStrategy Web server machine via Hypertext Transfer Protocol (HTTP). The request will be handled by Microsoft IIS, which will call the MicroStrategy Web. MicroStrategy Web will generate a report request and pass the request to the MicroStrategy Intelligence Server. The request will then create a report request that will be passed to the Informatica Metadata via an Open Database Connectivity (ODBC) connection. Query results will be passed back to MicroStrategy Intelligence Server, MicroStrategy Web, Microsoft IIS, and finally back to the client browser.

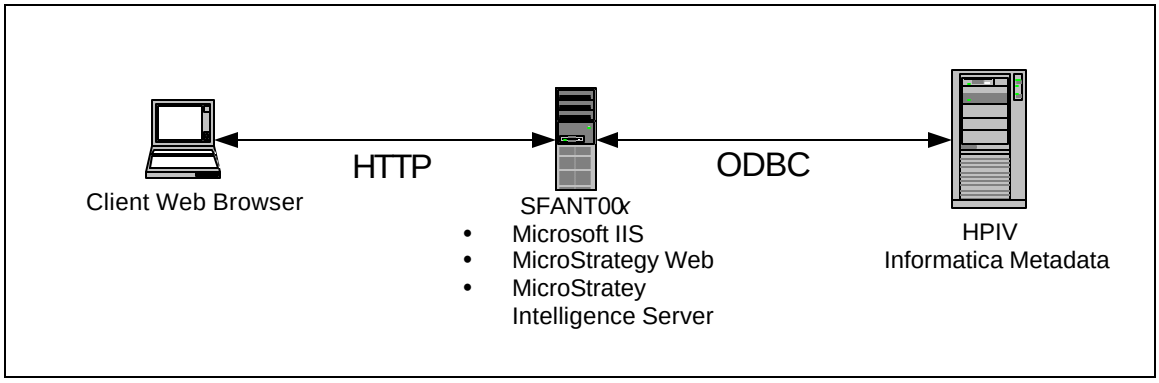

Figure 2 – Test Scenario # 1 – MicroStrategy – Detail Diagram

3.1.4. Test Scenario Inputs

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 6 – Test Scenario # 1 – MicroStrategy – Procedure Steps

#	User Action	Expected Observable Result
1.	The URL to the MicroStrategy Web project will be entered into a user browser: **4.20.14.245/MicroStrategy7/default.asp**	The MicroStrategy Web Welcome page will appear within the web browser.
2.	Left click on the word **Metadata**	The MicroStrategy Web Login page will appear within the web browser.
3.	The user will need to login to the project using the following login / password combination: Login ID: **test** Password: *password* Press the button labeled **Login**	The VMall project home page will appear within the web browser.

#	User Action	Expected Observable Result
4.	Left click on the words **Shared Reports**	The list of reports will appear within the web browser.
5.	Select report **Attribute ETL Lookup**	A report prompt will appear within the web browser.
6.	Select **Days Delinquent**	The report will begin generating. A page titled, "Your request has been submitted; Please wait…" will be displayed followed by the display of the report in grid format.

3.1.5. Test Scenario Expected Results

The output of the test will be report results displayed through the users web browser.

- The report results should identify the Extract, Transform and Load (ETL) information for the selected attribute, Days Delinquent.
- The ETL information should be displayed in the format shown in the screenshot which follows, and the data should be as outlined below:

Field Name: Target Column Name	Expected Result: MONTHLY_PAYMENT_AMT
Field Name: Source Field Name	Expected Result: MTH_PAY_AMT
Field Name: Source Name	Expected Result: DDRFILE_LOAN_TYPE
Field Name: Target Name	Expected Result: FACT_DELINQUENCY
Field Name: Transformation Expression	Expected Result: :SD.DDRFILE_LOAN_TYPE.MTH_PAY_AMT
Field Name: Comments	Expected Result: This mapping is the key mapping for populating the Fact Delinquency table and the associated Borrower / Student tables.
Field Name: Mapping Date	Expected Result: 10/04/2000 10:51:54

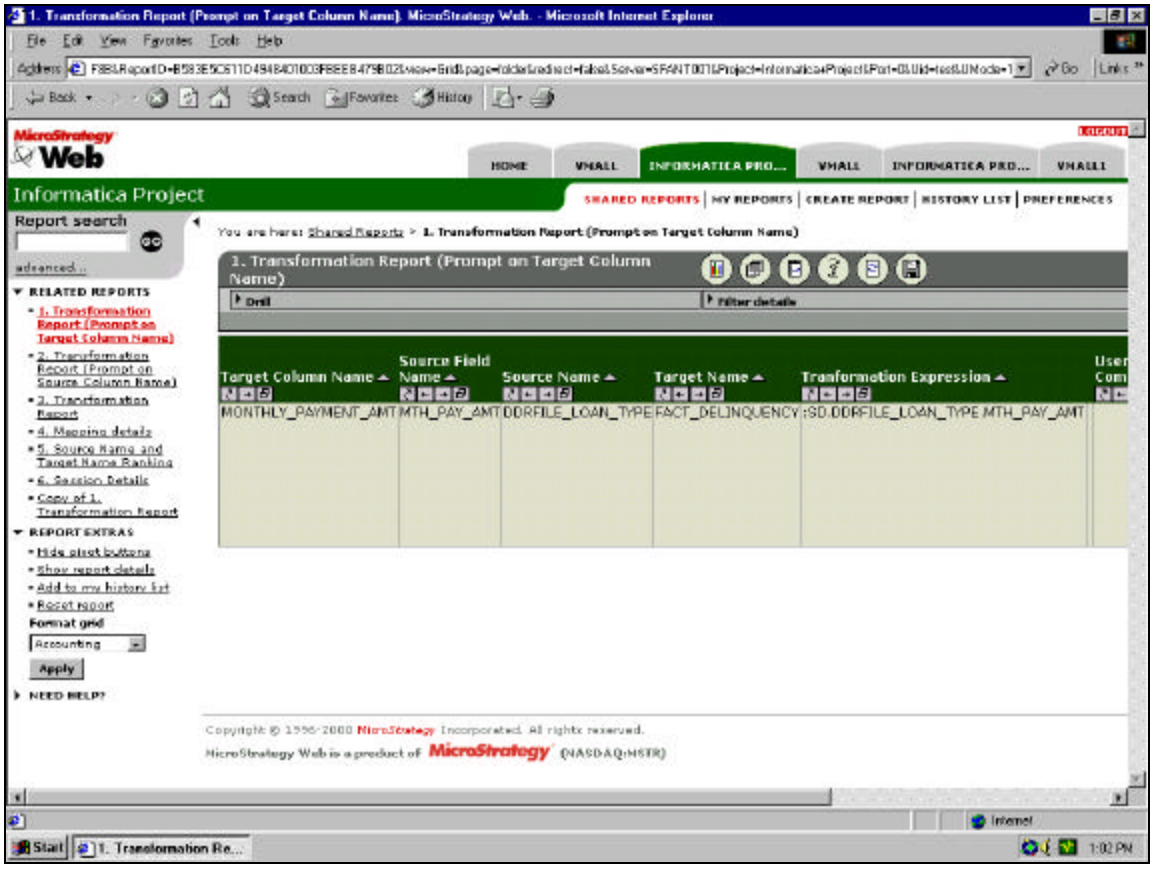

Figure 3 – Test Scenario # 1 – MicroStrategy – Report Generation Screenshot

3.1.6. Test Scenario Acceptance Criteria

The acceptance criteria will be as follows:

- The generation of a report without any errors or warnings.
- The display of the report results via the browser is identical to the ETL information retrieved directly from the Informatica metadata repository as shown in the Expected Results section above.

3.2. Scenario # 2 – MicroStrategy – OLAP Analysis over the Web

3.2.1. Test Scenario Description

This test will demonstrate that a user can generate a report from the data warehouse through a web browser. The purpose of this test is to verify the functionality of the web Relational On-Line Analytical Processing (ROLAP) services that are provided by the data warehouse architecture. The following components will be used in the performance of this test:

- MicroStrategy Web
- MicroStrategy Intelligence Server
- Microsoft IIS
- Oracle Data Warehouse

3.2.2. Test Scenario Dependencies and Required Resources

Table 7 – Test Scenario # 2 – MicroStrategy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		A connection to the NT Server which holds MicroStrategy Web must be available from the user location (SFA LAN)
Personnel		✓	User
Test Data	✓	✓	VMall warehouse and VMall MicroStrategy Metadata.
Prior Testing		✓	Validation of the configuration and operation of the eNetwork Dispatcher cluster for MicroStrategy
Hardware		✓	See Deliverable 16.1.2 Volume 4 - Data Warehouse Architecture
Software		✓	See Deliverable 16.1.2 Volume 4 - Data Warehouse Architecture
Time to be Allotted			5 minutes.

3.2.3. Test Scenario Detailed Design Description

1. Using a web browser, a user will connect to the MicroStrategy Web server machine via HTTP and generate a report request.
2. The request will be handled by Microsoft IIS, which will call the MicroStrategy Web.
3. MicroStrategy Web will generate a report request and pass the request to the MicroStrategy Intelligence Server.

4. The MicroStrategy Intelligence Server will then create a report request that will be passed to the data warehouse via an ODBC connection.

5. Query results will be passed back to MicroStrategy Intelligence Server, MicroStrategy Web, Microsoft IIS, and finally back to the client browser.

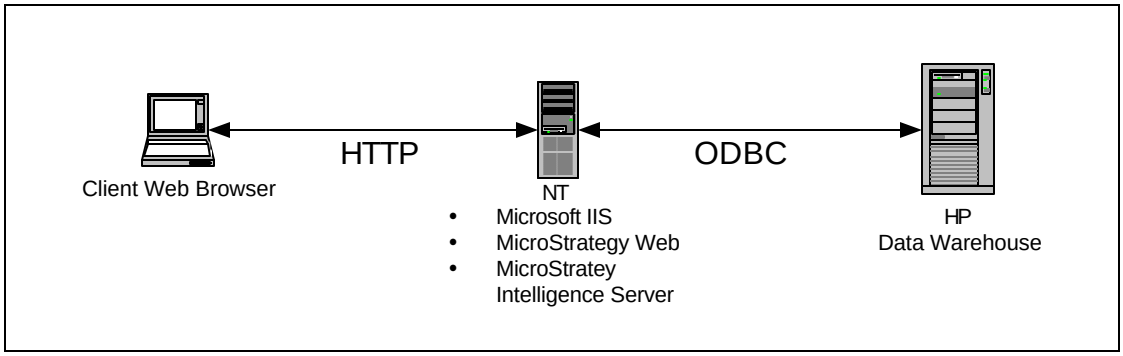

Figure 4 – Test Scenario # 2 – MicroStrategy – Detail Diagram

3.2.4. Test Scenario Inputs

Input to the test is the MicroStrategy demonstration data warehouse, VMall in the Oracle database.

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 8 – Test Scenario # 2 – MicroStrategy – Procedure Steps

#	User Action	Expected Observable Result
1.	The URL to the MicroStrategy Web project will be entered into a user browser: **4.20.5.244/MicroStrategy7/default.asp**	The MicroStrategy Web Welcome page will appear within the web browser.
2.	Left click on the word **VMall**	The MicroStrategy Web Login page will appear within the web browser.
3.	The user will need to login to the project using the following login / password combination: Login ID: **test** Password: *password* Press the button labeled **Login**	The VMall project home page will appear within the web browser.
4.	Left click on the words **Shared Reports**	A list of report folders will appear within the web browser

#	User Action	Expected Observable Result
5.	Left click on the words **b. Reports by Application**	A list of reports will appear within the web browser
6.	Left click on the words **2. Sales by Catalog**	The report will begin generating. A page titled, "Your request has been submitted; Please wait…" will be displayed followed by the display of the report in grid format.

3.2.5. Test Scenario Expected Results

Report View:

Catalog	Metrics	Dollar Sales
Spring 97		$ 86,714
Summer 97		$ 129,216
Fall 97		$ 104,435
Winter 97		$ 100,953
Spring 98		$ 104,587
Summer 98		$ 104,665
Fall 98		$ 94,210
Winter 98		$ 79,289

3.2.6. Test Scenario Acceptance Criteria

The acceptance criteria will be as follows:

- The generation of the two reports without any errors or warnings.
- The report results are identical to the results above.

3.3. Scenario # 3 – MicroStrategy – Server Fail Over

3.3.1. Test Scenario Description

This test will demonstrate that a user will not lose the ability to perform On-Line Analytical Processing (OLAP) Web analysis if the primary MicroStrategy Web / MicroStrategy Intelligence Server machine stops functioning. The secondary MicroStrategy Web / MicroStrategy intelligence Server machine will handle all functionality without an administrator's intervention. The purpose of this test is to verify MicroStrategy's fail over capabilities. The following components will be used in the performance of this test:

- MicroStrategy Web
- MicroStrategy Intelligence Server
- Microsoft IIS
- International Business Machines (IBM) Network Dispatcher (ND)
- Oracle Data Warehouse

3.3.2. Test Scenario Dependencies and Required Resources

Table 9 – Test Scenario # 3 – MicroStrategy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		A functioning, web enabled MicroStrategy project must be available (see test #1: OLAP analysis over the Web)
Personnel		✓	User NT administrator located at the Virtual Data Center (VDC) (needed to turn off the primary NT Server machine)
Test Data	✓	✓	VMall warehouse and VMall MicroStrategy Metadata.
Prior Testing	✓		Validation of the configuration and operation of the eNetwork Dispatcher cluster for MicroStrategy
Hardware		✓	See Deliverable 16.1.2 Volume 4 - Data Warehouse Architecture
Software	✓		See Deliverable 16.1.2 Volume 2 - Internet Architecture
Time to be Allotted			10 minutes

3.3.3. Test Scenario Detailed Design Description

Using a web browser, a user will make a request to the MicroStrategy Web server machine via http. This request will be handled by Network Dispatcher, which will forward the request

to the primary MicroStrategy Web / Intelligence Server machine (SFANT002). The primary MicroStrategy Web / intelligence Server machine will handle the request, submit report requests to the data warehouse via ODBC, and send the results back to the user's browser.

After this is complete, the primary MicroStrategy Web / Intelligence Server machine (SFANT002) will be turned off. The user will make a second report request to the MicroStrategy Web server machine. The request will be handled by Network Dispatcher, which will forward the request to the secondary MicroStrategy Web / Intelligence Server machine (SFANT003). This secondary MicroStrategy Web / Intelligence Server machine will handle the request, submit report requests to the data warehouse via ODBC, and send the results back to the user's browser.

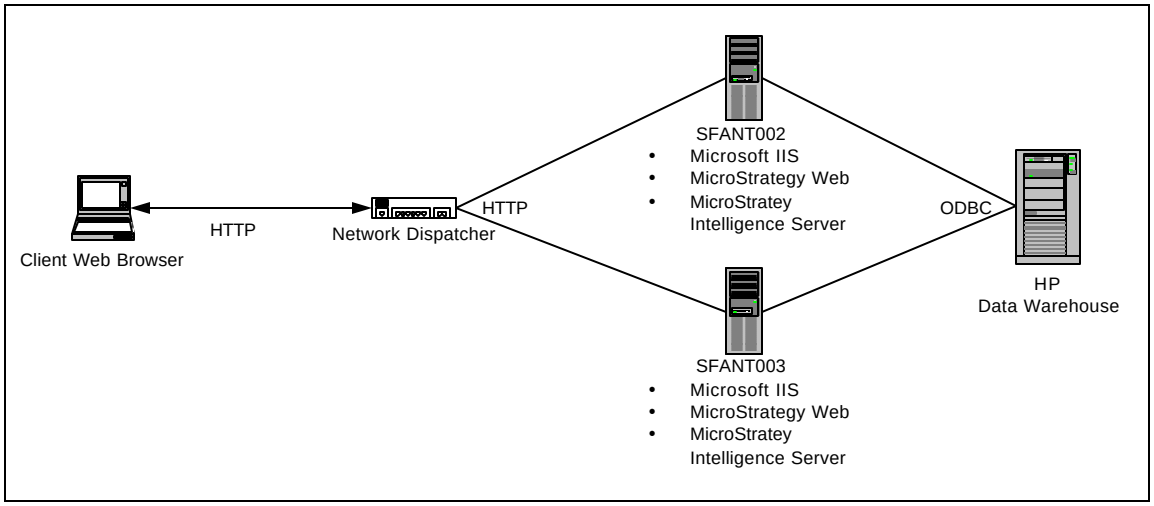

Figure 5 – Test Scenario # 3 – MicroStrategy – Detail Diagram

3.3.4. Test Scenario Inputs

Inputs to the test are a web enabled MicroStrategy Project (VMall) and a process that will stop functionality on the primary MicroStrategy Web / MicroStrategy Intelligence Server machine.

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 10 – Test Scenario # 3 – MicroStrategy – Procedure Steps

#	User Action	Expected Observable Result

#	User Action	Expected Observable Result
1.	The URL to the MicroStrategy Web project will be entered into a user browser: **4.20.14.245/MicroStrategy7/default.asp**	The MicroStrategy Web Welcome page will appear within the web browser.
2.	Left click on the word **VMall**	The MicroStrategy Web Login page will appear within the web browser.
3.	The user will need to login to the project using the following login / password combination: Login ID: **test** Password: *password* Press the button labeled **Login**	The VMall project home page will appear within the web browser.
4.	Left click on the words **Shared Reports**	A list of report folders will appear within the web browser
5.	Left click on the words **b. Reports by Application**	A list of reports will appear within the web browser
6.	Left click on the words **2. Sales by Catalog**	The report will begin generating. A page titled, "Your request has been submitted; Please wait…" will be displayed followed by the display of the report in grid format.
7.	Once the report is completed; the primary MicroStrategy Web / MicroStrategy Intelligence Server NT Server machine should be turned off	Nothing.
8.	From the list of reports on the left side of the screen, left click on the words **4. 98 Quarterly Sales.**	The following error will occur: *Error in Login: MicroStrategy Server error: MSIXML: server <server name> not connected.*
9.	The URL to the MicroStrategy Web project will be entered into a user browser: **4.20.14.245/MicroStrategy7/default.asp**	The MicroStrategy Web Welcome page will appear within the web browser.
10.	Left click on the word **VMall**	The MicroStrategy Web Login page will appear within the web browser.
11.	The user will need to login to the project using the following login / password combination: Login ID: **test** Password: *password* Press the button labeled **Login**	The VMall project home page will appear within the web browser.
12.	Left click on the words **Shared Reports**	A list of report folders will appear within the web browser
13.	Left click on the words **b. Reports by Application**	A list of reports will appear within the web browser
14.	Left click on the words **4. 98 Quarterly Sales**	The report will begin generating. A page titled, "Your request has been submitted; Please wait…" will be displayed followed by the display of the report in grid format.

3.3.5. Test Scenario Expected Results

The output of the test will be 2 report results. The first will be the results for *2. Sales by Catalog* generated form the primary MicroStrategy Web / MicroStrategy Intelligence Server machine. The second will be the results for *4. 98 Quarterly Sales* generated from the secondary or fail over MicroStrategy Web / MicroStrategy Intelligence Server machine. The report results are as follows:

Sales by Catalog:

Catalog	Metrics	Dollar Sales
Spring 97		$ 86,714
Summer 97		$ 129,216
Fall 97		$ 104,435
Winter 97		$ 100,953
Spring 98		$ 104,587
Summer 98		$ 104,665
Fall 98		$ 94,210
Winter 98		$ 79,289

98 Quarterly Sales:

Quarter	Metrics	Dollar Sales	Last Year's Dollar Sales
Q1 1998		$46,525	$21,755
Q2 1998		$17,954	$16,846
Q3 1998		$25,034	$20,279
Q4 1998		$29,262	$24,832

3.3.6. Test Scenario Acceptance Criteria

The acceptance criteria will be as follows:

- The generation of the two reports without any errors or warnings.
- The user is able to generate the second report (98 Quarterly Sales) with little interruption and no administrator intervention.

3.4. Scenario # 4 – Interwoven – Push Content to TeamSite Staging Repository (Internal Test)

3.4.1. Test Scenario Description

This test will start by having an existing document that will be processed through the TeamSite workflow process and placed into the TeamSite staging repository.

3.4.2. Test Scenario Dependencies and Required Resources

Table 11 – Test Scenario # 4 – Interwoven – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		FTP access to su35e4.ed.gov Connection to EdLAN
Personnel		✓	User
Test Data	✓	✓	Web page either on a floppy disk or on the workstation with known content
Prior Testing			Not Applicable
Hardware		✓	Su35E4
Software	✓		A web browser either IE 5.0 or Netscape 4.51 and higher
Time to be Allotted			10 minutes

3.4.3. Test Scenario Detailed Design Description

The detail diagram below shows the connectivity that will take place during this test. Launchpad and Samba are used if you edit the file or use Windows Explorer to view/copy data over to the TeamSite repository. The iwproxy is used when a page may have Server Side Includes (SSI) in order to view the page.

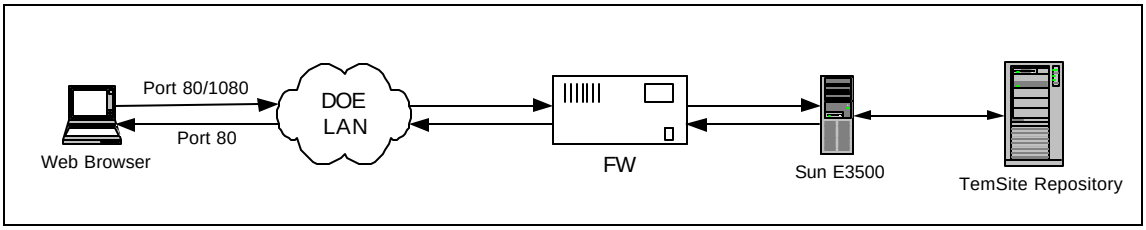

Figure 6 – Test Scenario # 4 – Interwoven – Detail Diagram

The above diagram shows information going through the Department of Education's LAN. The computer is accessing the TeamSite server through a browser throughout this test.

Samba is being used for Authentication and to see the content on a drive (Y:) connected to the workstation.

3.4.4. Test Scenario Inputs

The input is the content of a web page that will be created by the user and placed into the TeamSite server.

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 12 – Test Scenario # 4 – Interwoven – Procedure Steps

#	User Action	Expected Observable Result
1.	Start -> Run Type: **ftp su35e4.ed.gov**	An ftp window will appear asking for a username.
2.	Login ID: **ts** Password: *password*	If login is correct the next line will verify and you will be at the ftp prompt.
3.	At the ftp> prompt, type: **cd /iwmnt/default/main/viadortest/WORKAREA/OpenDeploytest**	If typed in correctly the next line will state command successful.
4.	At the ftp> prompt, type: **lcd** *directorywherethefileislocated*	Again if correct "command successful".
5.	At the ftp> prompt, type: **put test.html**	If correct "command successful" and "transfer complete".
6.	At the ftp> prompt, type: **Quit**	The ftp window will disappear.
7.	Double click on browser icon	Browser will launch.
8.	In address field type: **SU35E4.ED.GOV/IW/**	The TeamSite login screen will appear.
9.	User will log into TeamSite <select> Administrator Login ID: **ts** Password: *password*	A screen will appear showing any tasks assigned to the Administrator.

#	User Action	Expected Observable Result
10.	Select [WORKAREA] button	Screen will change to show all Workareas off of the main branch.
11.	On the left frame click on Viadortest -> OpenDeploytest	The frame on the right should show all files in that Workarea.
12.	Scroll through the directory listing until the file that was copied into the workarea is found. Select the file by clicking in the checkbox	A Checkmark will appear in the box.
13.	View the file to make sure it looks correct by clicking on the file to view the new web page	A browser window appears showing the file as it would appear in a browser.
14.	In the new browser window select: File -> close	The above window will disappear.
15.	Go to the file menu option and select submit direct.	A screen pops up with 3 different windows.
16.	You can fill them out as you like and click on **Submit** button when finished.	If it was submitted correctly the previous screen is displayed with the file unchecked.
17.	Log out of TeamSite by clicking the LOGOUT button (on the bottom / left of window)	The TeamSite login screen will appear. Note: Date _____ Note: Time _____

3.4.5. Test Scenario Expected Results

The new document will show up in the TeamSite Graphical User Interface (GUI) in the directory / location in which the user placed it. It should be viewable within TeamSite and the Author can submit content.

3.4.6. Test Scenario Acceptance Criteria

The test is validated when the content that was created for this branch is activated.

3.5. Scenario # 5 – Interwoven – Push Content to Production System (External Test)

3.5.1. Test Scenario Description

This test will start by having an existing document or documents in TeamSite staging area. The purpose of this test scenario is to verify that the content available in the staging repository can be deployed.

3.5.2. Test Scenario Dependencies and Required Resources

Table 13 – Test Scenario # 5 – Interwoven – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN
Personnel		✓	User with a valid login into the TeamSite as administrator
Test Data	✓		Document from Test Scenario # 4 - Interwoven – Push Content to TeamSite Staging Repository (Internal Test)
Prior Testing	✓		Test Scenario # 4 – Interwoven – Push Content to TeamSite Staging Repository (Internal Test)
Hardware		✓	TeamSite Server
Software		✓	A web browser either IE 5.0 or Netscape 4.51 or higher
Time to be Allotted			10 minutes

3.5.3. Test Scenario Detailed Design Description

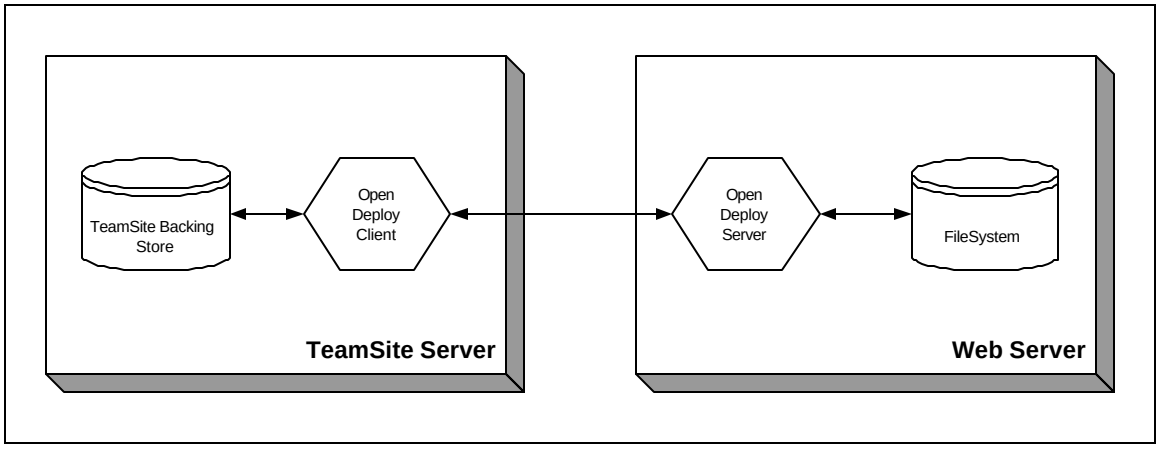

Figure 7 – Test Scenario # 5 – Interwoven – Detail Diagram

3.5.4. Test Scenario Inputs

Content to be deployed residing in the TeamSite staging repository.

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 14 – Test Scenario # 5 – Intewoven – Procedure Steps

#	User Action	Expected Observable Result
1.	Double click on browser icon	Browser window will appear.
2.	In address field type: **su35e4.ed.gov/iw/**	The TeamSite login screen will appear.
3.	log into TeamSite <select> Master Login ID: **ts** Password: *password*	If login was successful you will see the TeamSite GUI come up rather then the login screen.
4.	Select [WORKAREA] button	Screen should change to show all the main branches and information about the main branch.
5.	On the left window click Viadortest -> STAGING	The window on the right should show all the files in that branch, including the file you entered in the previous test.
6.	Verify file is there and content is correct by clicking on the file	A browser window should appear with the content displayed.
7.	In the new browser window select: File -> close	The above window will disappear.
8.	Click on the checkbox next to the file(s) you wish to deploy	A check mark should appear next to the file(s) selected.
9.	Go to File -> (scroll down) schools	A window is displayed telling how many bytes were sent. This amount should equal the size of the files sent. If it is zero then nothing was sent.
10.	Click on close window icon on the upper right hand corner	The OpenDeploy information window should disappear.
11.	Go to Start -> Run Type: **telnet test.schoolsportal.ed.gov**	Telnet window opens and displays login prompt.
12.	Login with id that can navigate the web content	
13.	At the prompt, type: **cd /opt/IBMHTTPD/htdocs/infospc/websqlbr/user/viador/doc**	

#	User Action	Expected Observable Result
14.	**ls –l test.html**	-rwxr-xr-x www 10:00 test.html NOTE: the above file will have the time that the file was actually staged in the previous test scenario.
15.	**cat test.html**	Contents of the file will be displayed. Examine the body of the file to verify that it matches the content from the previous test scenario.

3.5.5. Test Scenario Expected Results

The file will be on the production server in the web content directory on that machine.

3.5.6. Test Scenario Acceptance Criteria

The file pushed to the production server will be identical as the source file from the TeamSite staging repository.

3.6. Scenario # 6 – Viador – Portal Authentication

3.6.1. Test Scenario Description

The Portal Authentication scenario will demonstrate authentication services provided by the Viador Portal Framework.

3.6.2. Test Scenario Dependencies and Required Resources

Table 15 – Test Scenario # 6 – Viador – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN Viador Oracle database repository
Personnel		✓	User
Test Data		✓	A valid user id and password in the Viador repository
Prior Testing			N / A
Hardware	✓		Viador Server and the Viador Data Repository
Software		✓	Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			5 min

3.6.3. Test Scenario Detailed Design Description

1. The client submits user credentials (username and password) to the web server with a post method.

2. The web server directs the request to the servlet engine.

3. The servlet engine forwards the request to the Viador Information Center (VIC).

4. The VIC launches the Authentication Portlet through Application Programming Interface (API) calls that access the repository to verify the username and password supplied by the user.

5. If the supplied username and password matches the repository username and password, a portal session is established and the portal Launch Page is displayed for the user.

6. If the supplied username and password does not match the repository username and password, a “username or password is invalid” message is displayed in the user’s Web Browser.

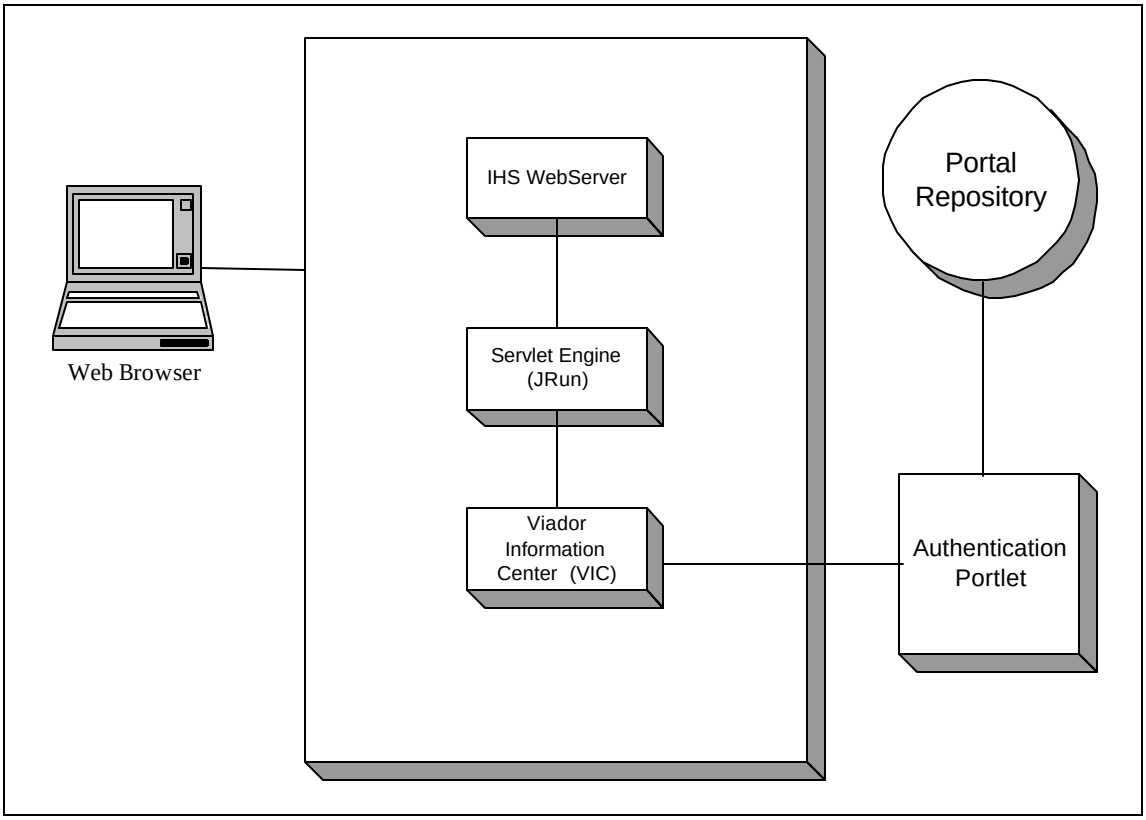

Figure 8 – Test Scenario # 6 – Viador – Detail Diagram

3.6.4. Test Scenario Inputs

The Portal Authentication scenario will require the following input from an existing portal user account:

Username: **techarch**

Password**:** *password*

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

Table 16 – Test Scenario # 6 – Viador – Procedure Steps

#	User Action	Expected Observable Result
1.	Launch the web browser program by double clicking on the browser icon seen on the desktop	The Web Browser window will display.

#	User Action	Expected Observable Result
2.	Supply the following URL to the browser, **http://su35e1.ed.gov:8181/infospc/index.html**	The Viador log-in Page should display.
3.	Enter the following username and password in the Log-In page and press the login button. Username: **techarch** Password: *password*	The Portal Launch page appears in a separate browser window.

3.6.5. Test Scenario Expected Results

Upon successful authentication, the Portal Launch Page should be displayed in the users Web Browser with a welcome string denoting the full name assigned to the user account upon creation.

3.6.6. Test Scenario Acceptance Criteria

The Portal Authentication scenario is accepted once the Portal Launch Page is displayed in the users Web Browser.

3.7. Scenario # 7 – Viador – Portal Socket Search

3.7.1. Test Scenario Description

The portal socket search scenario will demonstrate the Portals ability to perform a search through a Java socket connection using the Autonomy search engine.

3.7.2. Test Scenario Dependencies and Required Resources

Table 17 – Test Scenario # 7 – Viador – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		IP and port access to Autonomy DRE. The Autonomy Server should be up and running. Viador Oracle database repository
Personnel		✓	User
Test Data		✓	Query String
Prior Testing	✓		Viador Authentication scenario
Hardware	✓		Viador and Autonomy servers
Software		✓	Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			15 minutes

3.7.3. Test Scenario Detailed Design Description

1. A user submits a search query through the web browser.
2. The web browser sends the request to the web server.
3. The web server sends the request to the servlet engine.
4. The servlet engine passes the request to the VIC.
5. The Viador server invokes the Search Portlet.
6. The Search Portlet establishes a socket connection with the Autonomy server and forwards the search query to the Autonomy Data Reasoning Engine (DRE).
7. The DRE executes the query and returns the search results to the portlet as a HTTP string.
8. The portlet displays the search results in the user's Web Browser.

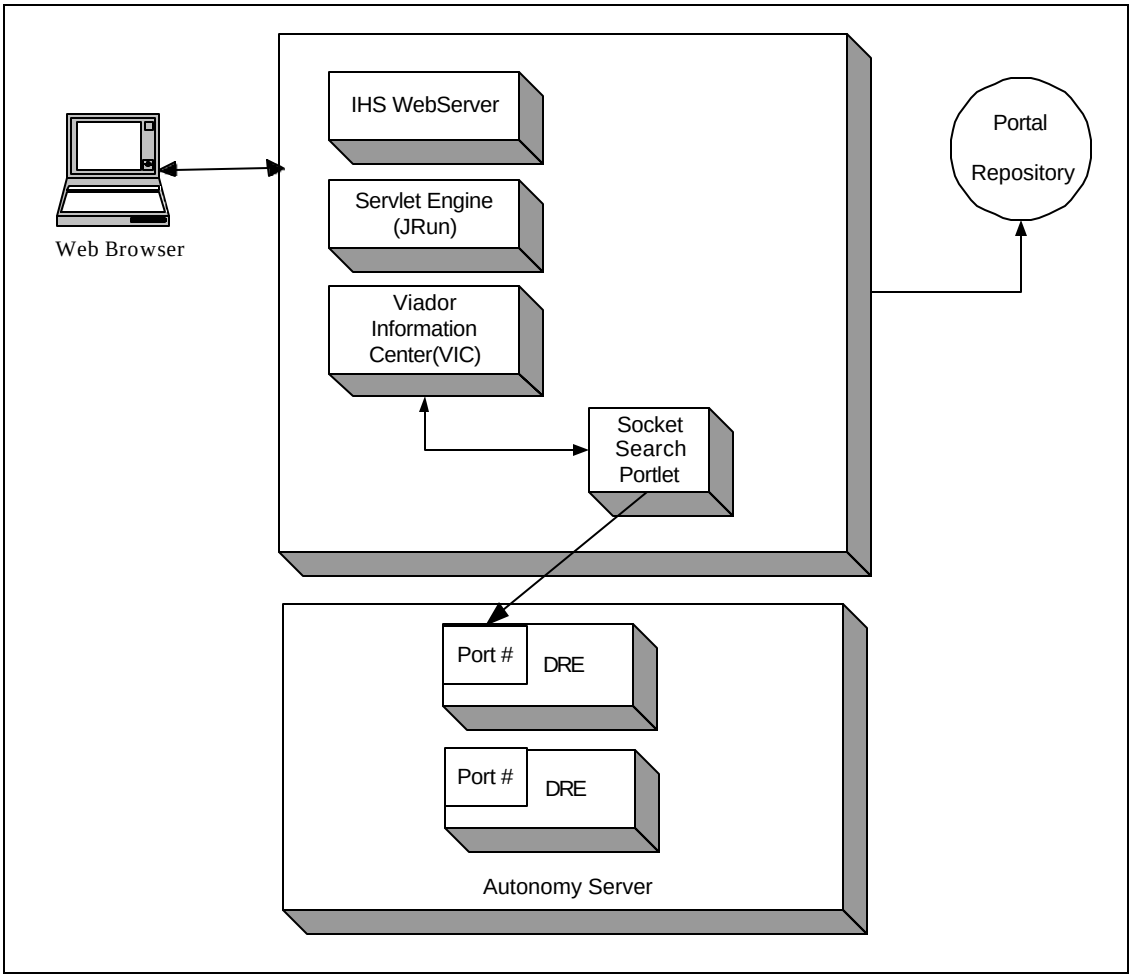

Figure 9 – Test Scenario # 7 – Viador – Detail Diagram

3.7.4. Test Scenario Inputs

The Portal socket search scenario requires the successful execution of the Portal Authentication scenario.

The Portal socket search scenario requires a search query to be supplied to the search form.

Test Procedure Steps

Table 18 – Test Scenario # 7 – Viador – Procedure Steps

#	User Action	Expected Observable Result
1.	Launch the web browser program by double clicking on the browser icon seen on the desktop	The Web Browser window will display.

#	User Action	Expected Observable Result
2.	Supply the following URL to the browser, **http://su35e1.ed.gov:8181/infospc/index.html**	The Viador portal login page appears in the browser.
3.	Enter the following username and password in the resulting page and press the login button. Username: **techarch** Password: *password*	The portal page will appear in the browser.
4.	Click on the search portlet icon	A form appears prompting the user to enter the search query.
5.	Supply search query to form and click the search button Query: **sports**	Search results for the query are displayed in the web browser.

3.7.5. Test Scenario Expected Results

Search results collected by the DRE are displayed through the user's Web Browser.

3.7.6. Test Scenario Acceptance Criteria

The Portal Search scenario is accepted when the user can view requested search results and it is validated that the results are consistent with the information stored in the Autonomy database.

3.8. Scenario # 8 – WebSphere – Application Server JDBC Request

3.8.1. Test Scenario Description

This test verifies the ability of the WebSphere Application Server (WAS) to connect to a database on a remote server through Java Database Connectivity (JDBC).

In this test, a user will enter parameter information into a Hypertext Markup Language (HTML) form that will trigger a WebSphere servlet to use JDBC to perform a simple count query on a database. For the purposes of testing the ITA architecture, the database engine will be Oracle 8i running on a Hewlett-Packard (HP) system. However, the same test should be applicable to any JDBC enabled database.

3.8.2. Test Scenario Dependencies and Required Resources

Table 19 – Test Scenario # 8 – WebSphere – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN
Personnel		✓	User
Test Data		✓	The tables which are in the database.
Prior Testing			None
Hardware	✓		The WebSphere, IHS and database servers must be configured and operational.
Software		✓	WebSphere must have the Oracle JDBC driver loaded. Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			5 minutes

3.8.3. Test Scenario Detailed Design Description

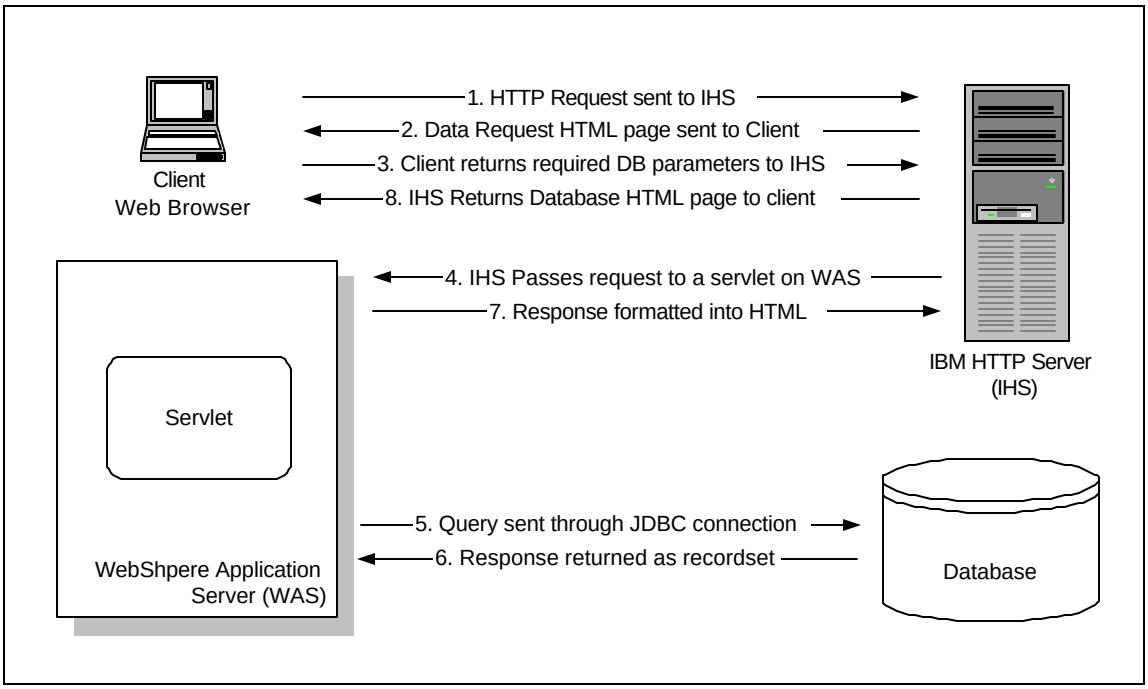

Figure 10 – Test Scenario # 8 – WebShpere – Detail Diagram

1. A user will request the HTML DB parameter page from the IBM HTTP Server (IHS) server.

2. The web server returns the requested page. This page provides input fields for the user to enter the required database connection variables: DB URL, User ID, and password.

3. The user enters the required inputs and an HTTP request is sent to IHS

4. IHS passes the request to a servlet running on WebSphere

5. The servlet creates a connection to the database through JDBC and queries the database for it's associated metadata.

6. The database returns a recordset object containing the metadata to the servlet.

7. The servlet formats the data into HTML and returns it to IHS.

8. IHS passes the associated data back to the client.

3.8.4. Test Scenario Inputs

To perform the test, all of the default parameters that are requested in the "WebSphere JDBC to Oracle" test must be valid. If any of these values are not valid the user can change them before pressing the submit button.

Table 20 – Test Scenario # 8 – WebSphere – Procedure Steps

#	User Action	Expected Observable Result
1.	Connect to the WebSphere testing main menu: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	Web page titled "WEB SERVER TESTING MAIN MENU" is displayed.
2.	Click "Go to JDBC Test Servlet" under the HTTP test menu	The web page titled "WebSphere JDBC to Oracle Test" is displayed.
3.	Verify that the Server Name, Port number, database name, user name, and database table are valid. If the values are not valid, make any necessary changes.	(none)
4.	Click the **Submit** button	The result page, titled "WebSphere JDBC to Oracle Test" is displayed. The text "The number of records in the users table is: 26" appears. If a different table name is used (or records have been added or deleted) the results will vary.

3.8.5. Test Scenario Expected Results

The text "The number of records in the users table is: 26" will appear on the resulting HTML page.

3.8.6. Test Scenario Acceptance Criteria

The test is considered valid if the database server returns a record count. This confirms that WebSphere has successfully used JDBC to communicate with the database server and retrieve the response to a query.

3.9. Scenario # 9 – WebSphere – Application Server Search

3.9.1. Test Scenario Description

The purpose of this test is to verify the ability of a servlet running in the WebSphere environment to query information from an Autonomy server and return the results to the user.

Autonomy Knowledge Server supports the cataloging of all types of documents into index files (called DRE's) which are then queried against to locate relevant information based on search criteria. Autonomy supports several interfaces for requesting information, including C, HTTP, and Java APIs. This test is designed to verify the ability of WebSphere to use the HTTP API for querying Autonomy based on specified search criteria. A web page will be developed to accept the required information from the user. The Submit button of the page will pass the parameters to IHS, which will call a WebSphere servlet, and send the request through Autonomy through the HTTP API. The response received from Autonomy will be formatted into an HTML document by the servlet, and then passed back through IHS to the user's browser.

3.9.2. Test Scenario Dependencies and Required Resources

Table 21 – Test Scenario # 9 – WebSphere – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		The WebSphere, IHS, and Autonomy servers must be installed, configured, running, and accessible through the SFA network
Personnel		✓	User
Test Data		✓	Query string
Prior Testing			None
Hardware	✓		The Autonomy server must be configured to receive requests through the HTTP API.
Software		✓	To perform the test using the system defaults, the Autonomy server must have a DRE built for the search keyword "test". Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			5 minutes

3.9.3. Test Scenario Detailed Design Description

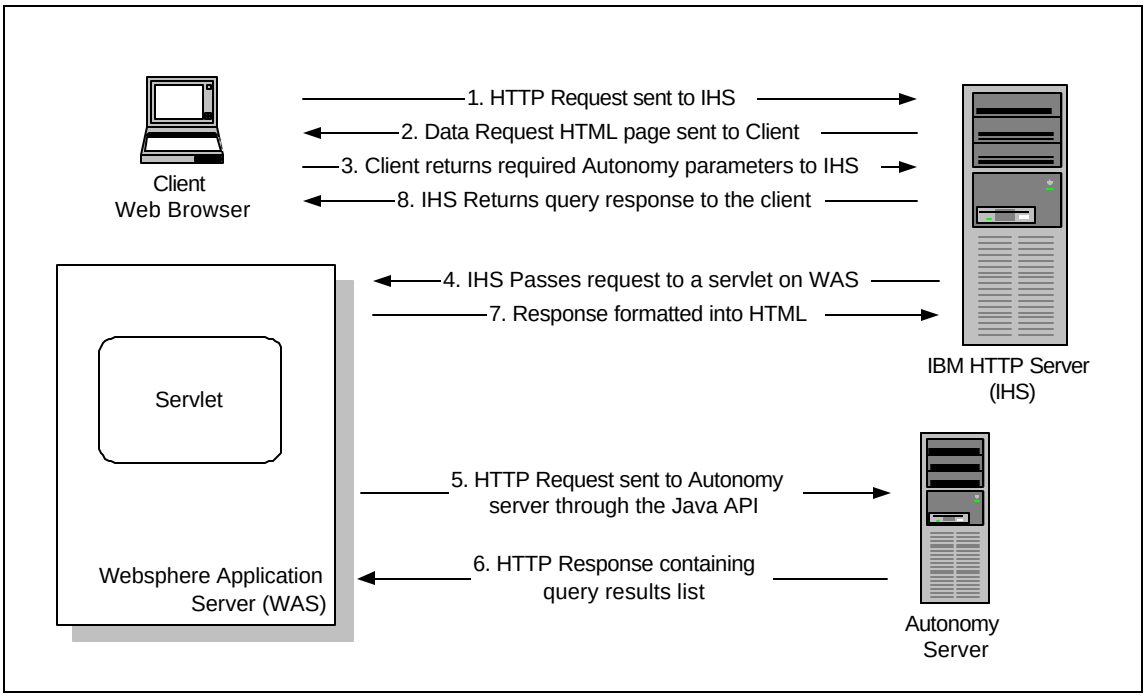

Figure 11 – Test Scenario # 9 – WebSphere – Detail Diagram

1 The client requests the HTML page used to submit requests to Autonomy.

2 IHS returns the requested web page.

3 The user fills in the relevant fields, and submits the request.

4 IHS forwards the request to a servlet on the WebSphere server.

5 The servlet uses the HTTP API to send a request to the Autonomy server based on the user's input.

6 The "query result list" is transferred by HTTP from the Autonomy server back to the servlet on the WebSphere server.

7 The servlet formats the response into an HTML document and transfers it back to IHS.

8 IHS sends the response to the client.

3.9.4. Test Scenario Inputs

To perform the test, the URL and Port number used by the Autonomy server must be known, and a search string that will return valid data must be obtained.

Table 22 – Test Scenario # 9 – WebSphere – Procedure Steps

#	User Action	Expected Observable Result
1.	Connect to the WebSphere testing main menu: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	Web page titled "Web Server Testing Main Menu" is displayed
2.	Click "Go to Autonomy" under the HTTP test menu	The web page titled "WebSphere Autonomy Test" is displayed
3.	Verify that the URL, Port number, and search criteria are valid. If the values are not valid, make any necessary changes.	(none)
4.	Click the **Submit Query** button	The result page, titled "Autonomy Response to Query" is displayed. Information regarding the response to the query appears on the page.

3.9.5. Test Scenario Expected Results

When valid input is provided, an HTML page will be displayed showing the number of matches, followed by name and summary information for each match. If the test fails, then the returned HTML document will show only the title of the page with no response information.

1.1.7 Test Scenario Acceptance Criteria

A response from the Autonomy server validates the test.

3.10. Scenario # 10 – WebSphere – Error Handling Report to User

3.10.1. Test Scenario Description

When a WebSphere Servlet or Java Server Page (JSP) operating within WebSphere generates an error, by default WebSphere will return an error message to the calling HTTP server. This message is then relayed to the user in an HTML document. The text of the document frequently contains specific information about the error as well as the program call stack that led up to the error.

This information may be useful to a software developer, however it is confusing to a system user. The solution is to define a general error page for WebSphere to return to the user when an error occurs. This page should have a "high level" error message which would be meaningful to the user as well as an e-mail address or phone number to contact in order to resolve the problem. This HTML page will be built by a servlet that will be called by WebSphere whenever an error occurs.

3.10.2. Test Scenario Dependencies and Required Resources

Table 23 - Test Scenario # 10 – WebSphere – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN
Personnel		✓	User
Test Data			None
Prior Testing			None
Hardware	✓		The WebSphere and IHS servers must be installed, configured, running, and accessible through the SFA network
Software		✓	The servlets and JSPs described below must be installed on the WebSphere server. Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			5 minutes

3.10.3. Test Scenario Detailed Design Description

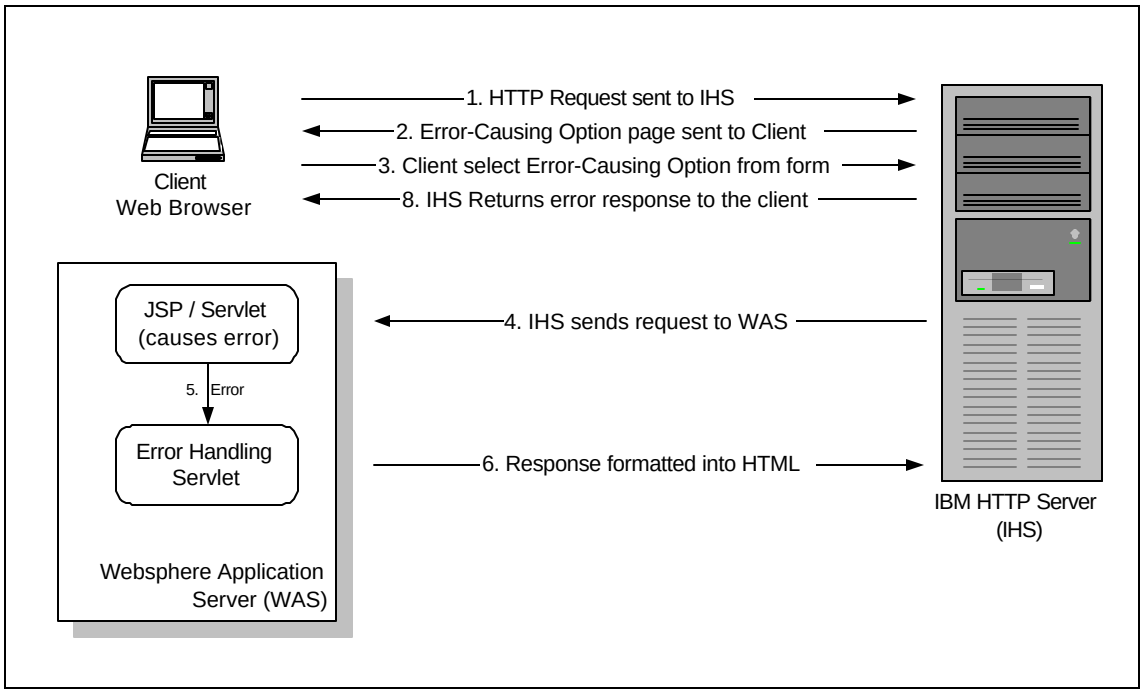

Figure 12 – Test Scenario # 10 – WebSphere – Detail Diagram

1. The client sends an HTTP request for the WebSphere testing page to IHS.
2. IHS returns the Error-causing HTML page to the user.
3. The client selects the “Error-causing Task” from the form.
4. The request is sent from IHS to WebSphere.
5. While attempting to complete the requested task, an error occurs on WebSphere.
6. The error handling servlet formulates an HTML response to the user and sends it to IHS.
7. IHS returns the generated HTML page to the user.

3.10.4. Test Scenario Inputs

Table 24 – Test Scenario # 10 – WebSphere – Procedure Steps

#	User Action	Expected Observable Result
1.	Connect to the WebSphere testing main menu: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	Web page titled WEB SERVER TESTING MAIN MENU is displayed.

#	User Action	Expected Observable Result
2.	Click **Go to Error Test** under the HTTP test menu	The web page titled "Error Test Options" is displayed.
3.	Select the **Servlet not Present** radio-button, and click the **submit** button	An HTML document will be returned containing a brief error message and contact information.
4.	Press the back button on your browser to return to the Error Test Options page	The web page titled "Error Test Options" is displayed again.
5.	Select the **JSP Version Error** radio-button, and click the **submit** button	An HTML document will be returned containing a brief error message and contact information.
6.	Press the **back** button on your browser to return to the Error Test Options page	The web page titled "Error Test Options" is displayed again.
7.	Select the **JDBC Error** radio-button, and click the **submit** button	An HTML document will be returned containing a brief error message and contact information.

3.10.5. Test Scenario Expected Results

The dynamically generated error handling HTML document will be returned to the client's web browser regardless of the error type that has occurred.

3.10.6. Test Scenario Acceptance Criteria

A response from the error handling servlet in each of the three error causing situations above validates this test. An "unfiltered" error returned by WebSphere to the user would indicate the failure of this test.

3.11. Scenario # 11 – IHS – Static Content Flow

3.11.1. Test Scenario Description

A client browser requests static content from IHS through a link.

3.11.2. Test Scenario Dependencies and Required Resources

Table 25 – Test Scenario # 11 – WebSphere – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN
Personnel		✓	User
Test Data		✓	Static HTML page
Prior Testing			None
Hardware	✓		The IHS server must be installed, configured, running, and accessible through the SFA network
Software		✓	Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above
Time to be Allotted			5 minutes

3.11.3. Test Scenario Detailed Design Description

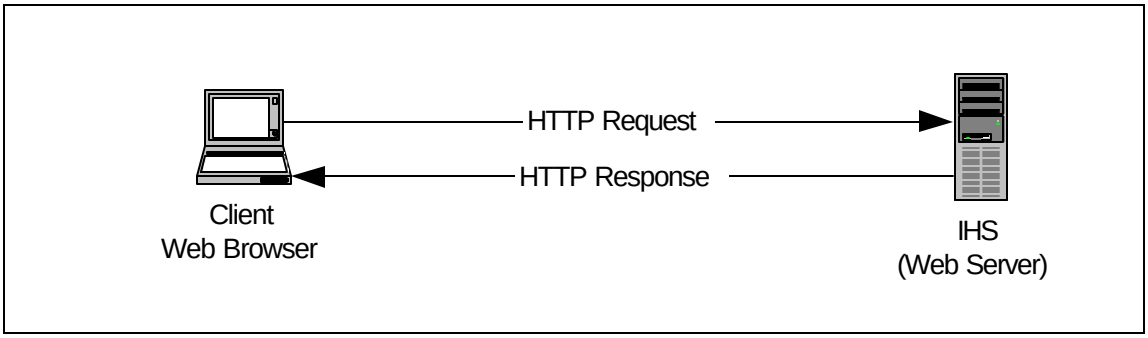

Figure 13 – Test Scenario # 11 – IHS – Detail Diagram

In this scenario, a browser requests content from a web sever. The server then returns a static HTML page to the browser. This scenario tests the functionality of IHS to provide static content.

1. Client (Browser) sends HTTP request to IHS.

2. IHS returns the static HTML page to the client.

3.11.4. Test Scenario Inputs

Test Procedure Steps

Table 26 – Test Scenario # 11 – WebSphere – Procedure Steps

#	User Action	Expected Observable Result
1.	In a web browser, enter the following URL: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	The MainMenu.htm page is displayed.
2.	Click "go to Static Page"	The following page is displayed: http://dev.ifap.ed.gov:8081/fsaHTML/StaticContent.htm

3.11.5. Test Scenario Expected Results

Browser received a static HTML page from IHS.

Returned URL: http://dev.ifap.ed.gov:8081/fsaHTML/StaticContent.htm

3.11.6. Test Scenario Acceptance Criteria

The client browser displays the requested static content from the web server.

3.12. Scenario # 12 – IHS – Dynamic Content Flows

3.12.1. Test Scenario Description

a. Servlet 2.1 – Browser requests URL, IHS passes to WAS, WAS triggers a servlet to deliver the response.

Testing: WAS Servlet support and Servlet Container.

b. JSP 1.0 – Browser requests URL with JSP extension, IHS passes to WAS, WAS triggers a servlet to deliver a response.

Testing: WAS JSP support and JSP compiler.

** (Test both 1 & 2 with HTTPS (Secure server))

Testing: Test HTTPS (secure server) support. Ensure HTTPS is set up correctly

3.12.2. Test Scenario Dependencies and Required Resources

Table 27 - Test Scenario # 12 – WebSphere – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems	✓		Connection to EdLAN
Personnel		✓	User
Test Data			None
Prior Testing		✓	Scenario # 11 – IHS – Static Content Flow
Hardware	✓		IHS and WAS installed with Servler2.1 and JSP1.0 enabled.
Software		✓	The servlets and JSPs described below must be installed on the WebSphere server. Minimum browser requirements: Internet explorer 4.0 and above or Netscape 4.5 and above **For Servlet created by using Servlet Builder in VAJ: Copy the ***ivjsb30.jar(for version 3.0, for version 2.0 use ivjsb20.jar)*** from C:\IBMVAJava\eab\runtime30 to C:\WebSphere\AppServer\lib\ivjsb30.jar. Within an X-Term session, launch the WebSphere Advanced Administrative Console. Once this has been successfully launched, add **"-classpath C:\WebSphere\AppServer\lib\ivjsb30.jar"** to the "**Command line arguments**" as shown in the following figure.
Time to be Allotted			5 minutes

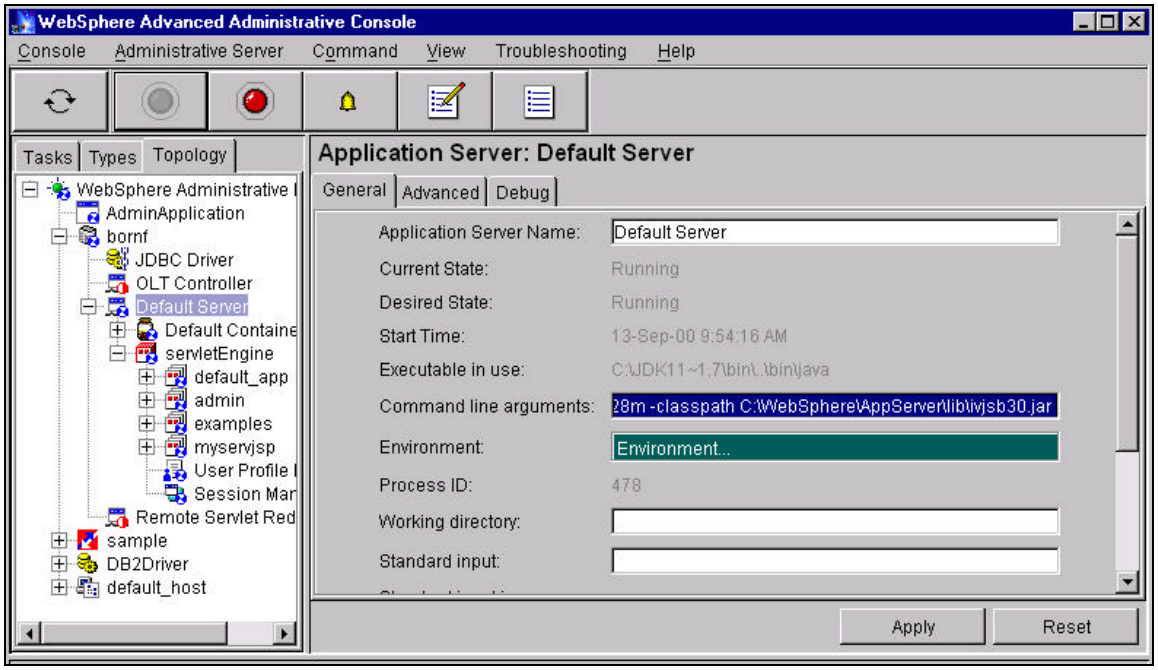

Figure 14 – Test Scenario # 12 – IHS – WebSphere Advanced Administrative Console

3.12.3. Test Scenario Detailed Design Description

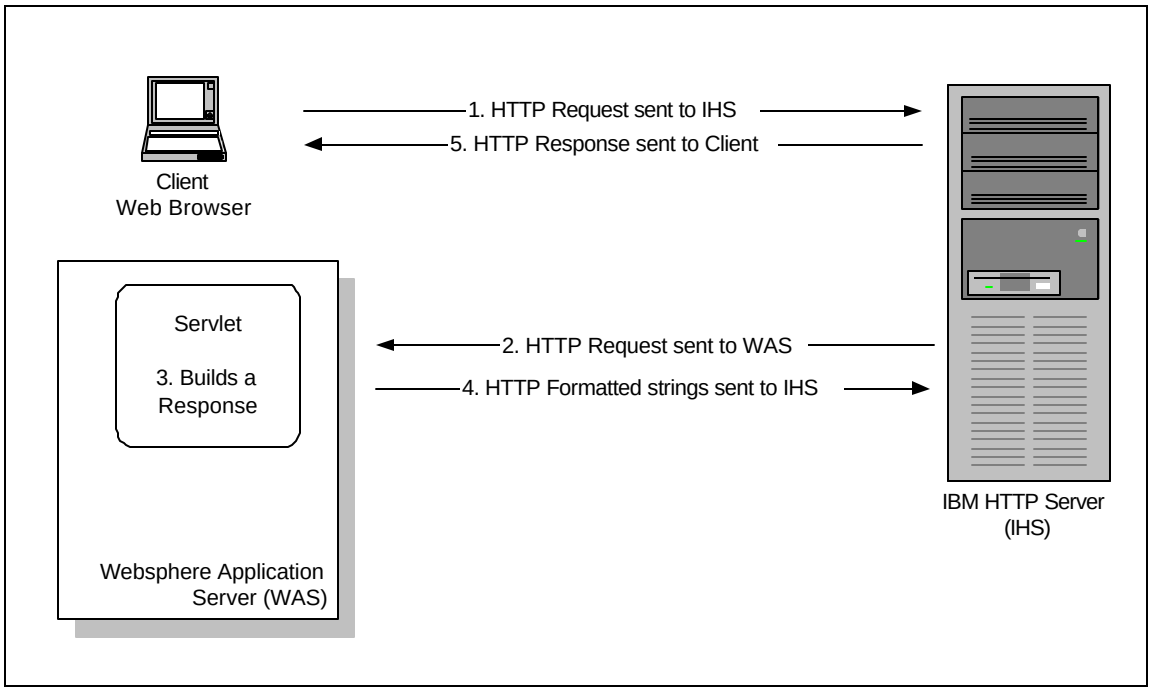

Figure 15 – Test Scenario # 12 – IHS – Detail Diagram A: Servlet

a. Servlet2.1- In this scenario when the server receives a request for a servlet it redirects the request to WAS. WAS then loads the correct servlet. The servlet then generates the HTML strings. The input to the servlet will be the HTTP request originally sent to the Web server by the client. The output from the servlet will be the HTML strings.

1. The client (browser) sends a request to the web server (IHS).
2. The server sends the request information to the servlet (runs inside WAS).
3. The servlet builds a response and passes it to IHS. The response is dynamically built, and the content of the response depends on the client's request.
4. IHS sends the response back to the client.

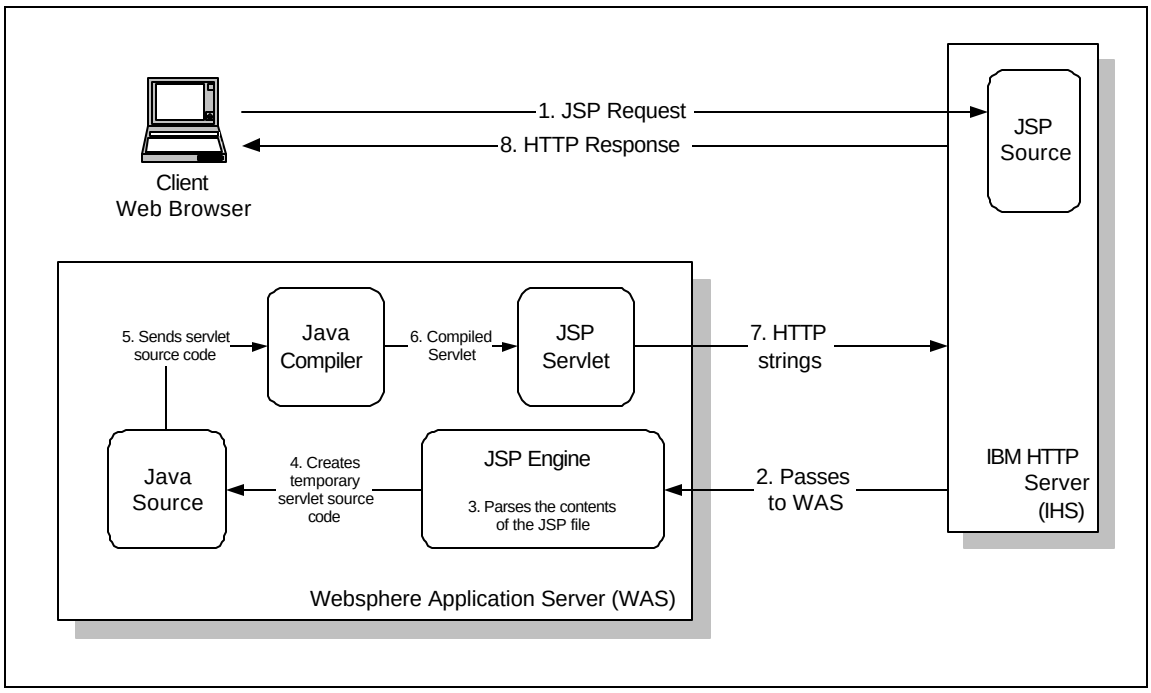

Figure 16 – Test Scenario # 12 – IHS – Detail Diagram B: JSP

b. JSP1.0- In this scenario when the server receives a JSP request, it detects the JSP tags and passes the JSP information to the JSP engine. The JSP engine then translates the contents of the JSP file into a servlet source code and passes it to the Java Compiler. The compiler then compiles the source code into servlet class file. WAS then instantiates and executes the servlet. The combination of static HTML and dynamic elements specified in the original JSP page definition are sent to the client through the output stream of the servlet's response object.

1. The client (browser) makes a JSP page request to the web server (IHS).
2. IHS passes the request to WAS.

3. JSP engine (in WAS) parses the contents of the JSP file.
4. JSP engine creates temporary servlet source code based on the contents of the JSP.
5. The Java compiler compiles the servlet source code into a servlet class file.
6. The servlet is instantiated. The init and service methods of the servlet are called, and the servlet logic is executed.
7. WAS returns the HTML page (generated from servlet) to the browser through IHS.

3.12.4. Test Scenario Inputs

Table 28 – Test Scenario # 12 – WebSphere – Procedure Steps

#	User Action	Expected Observable Result
1.	In a web browser, enter the following URL: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	The MainMenu.htm page is displayed.
2.	Click "go to servlet"	The following page is displayed: http://dev.ifap.ed.gov:8081/webapp/myservjsp/DynamicContent and contains the current date and time.
3.	In a web browser, enter the following URL: **http://dev.ifap.ed.gov:8081/fsaHTML/MainMenu.htm**	The MainMenu.htm page is displayed.
4.	Click "go to JSP"	The following page is displayed: http://dev.ifap.ed.gov:8081/webapp/myservjsp/JSPContent.jsp and contains (1) the current date and time and (2) number of times visited.

3.12.5. Test Scenario Expected Results

Browser received Dynamic HTML pages from IHS.

Servlet: Returns HTML page that contains current time and date information.

JSP: Returns HTML page that contains current time and date information and number of times visited.

3.13. Scenario # 13 – Autonomy – Internal Content Retrieval (File System)

3.13.1. Test Scenario Description

To retrieve content from the internal file system and put it into the DRE, Autonomy uses a process called autoindexer. This test ensures that the autoindexer functions as expected and content is retrieved from the local file system. The autoindexer needs to recognize new content files, import the content into an index file, and index the content into the DRE. This will also test that the autoindexer searches an entire directory structure recursively to find new content for the DRE.

3.13.2. Test Scenario Dependencies and Required Resources

Table 29 – Test Scenario # 13 – Autonomy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems			None
Personnel		✓	User
Test Data	✓		Test HTML files (With the current configuration, these files need to be on the Autonomy server and the web server in the same directory structure) /tmp/hello.htm /tmp/goodbye.htm
Prior Testing			None
Hardware		✓	Autonomy machine – SU35E2 Web server machine – SU35E3
Software		✓	Web browser, Autonomy Knowledge Server
Time to be Allotted			30 minutes

3.13.3. Test Scenario Detailed Design Description

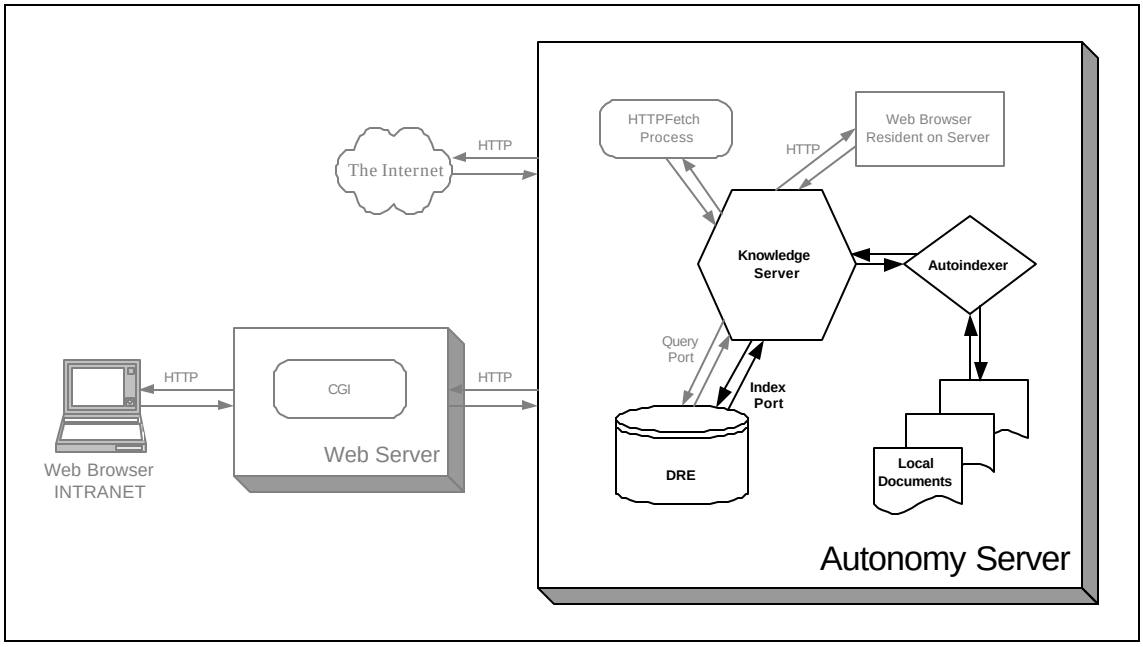

Figure 17 – Test Scenario # 13 – Autonomy – Detail Diagram

1. The autoindexer periodically polls the local file system for local documents specified in its configuration file.

2. When new local documents are found, the autoindexer indexes the documents.

3. The indexed documents are then imported to the Knowledge Server.

4. The Knowledge Server then imports the indexed documents into the DRE using the index port.

5. The DRE then sends a success or failure message through the Knowledge Server to the autoindexer, on whether or not the documents where successfully imported into the DRE.

3.13.4. Test Scenario Inputs

Test HTML files (With the current configuration, these files need to be on the Autonomy server and the web server in the same directory structure)

- /tmp/hello.htm
- /tmp/goodbye.htm

Table 30 – Test Scenario # 13 – Autonomy – Machine Ports

Environment	Data DRE Query Port	Data DRE Index Port	Community DRE Query Port	Community DRE Index Port	HTTPFetch Service Port	HTTPFetch Spider Port
Dev	60000	60001	60002	60003	60005	80
Tst	30000	30001	30002	30003	30005	80
Stg	40000	40001	40002	40003	40005	80

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

system* =** Replace this keyword with the appropriate environment abbreviation (e.g. /var/system***/autonomy is /var/dev/autonomy).

Table 31 – Test Scenario # 13 – Autonomy – Procedure Steps

#	User Action	Expected Observable Result
1.	Check if Knowledge Server process is running. **netstat –a | grep** *<query port #>*	If nothing is returned, then Knowledge Server process is not running; continue to step 2. If something is returned, skip to step 3.
2.	Start Knowledge Server process **cd /var/*system*/autonomy/autonomy_db/KnowServer210** **./StartQuery.sh** (inside script press [ENTER] twice) **ps –ef | grep queryh**	Knowledge Server process is started. The KnowServer210queryh process is shown in the process list.
3.	Launch a web browser **telnet su35e2** **cd /opt/netscape** export DISPLAY=<IP address of current machine>:0.0 **./netscape &**	Netscape web browser is displayed on the screen.

#	User Action	Expected Observable Result
4.	Ensure that the Knowledge Server is running and ready to index content from the autoindexer. Enter this address into the web browser to bring up the Knowledge Server Configuration page. **http://**<*autonomy machine IP*>**:**<data query port>**/qmethod=v**	The response from the browser indicates information about the Knowledge Server running. It will be of the form below: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 10000, Index port: 10001 0 documents 0 document sections 0 document slots 0 Terms Process ID: 24688 Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …
5.	Start the autoindexer of KnowServer210 on Autonomy server machine. **cd /var/*system*/autonomy/autonomy_db/KnowServer210** **./StartIndexer.sh** (inside script press [ENTER] twice) **ps –ef | grep autoi**	Knowledge Server autoindexer process is started. The KnowServer210autoindexer process is shown in the process list.
6.	Check the log file to make sure the autoindexer is running and waiting for something new to appear in its polling directory. **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer** **tail –2 KnowServer210autoindexer.log**	The autoindexer log file shows these lines: … Starting loop … … Job0 sleeping (no files matched/updated since last run)…
7.	Copy the test hello.htm file into the autoindexer polling directory. **cp /tmp/hello.htm /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/**	hello.htm is copied to /var/system/autonomy/autonomy_db/KnowServer210/autoindexer/files/ on Autonomy machine.

#	User Action	Expected Observable Result
8.	Copy hello.htm into the autoindexer polling directory. **telnet** <*web server machine*> **mkdir /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/** **exit** **ftp** <*web server machine*> **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/** **lcd /tmp** **put hello.htm** **quit**	hello.htm is copied to /var/system/autonomy/autonomy_db/KnowServer210/autoindexer/files/ on web server machine.
9.	The autoindexer should automatically recognize the hello.htm file in the polling directory and begin the process to import into the DRE. Look at the autoindexer log file again to verify that hello.html was imported and indexed. **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer** **tail KnowServer210autoindexer.log**	The autoindexer log should contain lines indicating that the IMPORT COMMAND and INDEXING COMMAND were successful.
10.	Check that the DRE contains the document added by the autoindexer. Bring up the Knowledge Server configuration page in the web browser. **http://**<*autonomy machine IP*>**:**<*data query port*>**/qmethod=v**	The browser indicates that one document was added to the DRE. It will be of the form below: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 60000, Index port: 60001 1 documents 1 document sections 1 document slots 4 Terms Process ID: 24688 Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …

#	User Action	Expected Observable Result
11.	Ensure that the autoindexer is configured to search directories recursively for new files. **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer** **vi KnowServer210autoindexer.cfg** Change or verify that the directoryRecursive=on under the [DEFAULT] section	[DEFAULT] … directoryRecursive=on …
12.	Copy the test goodbye.htm file into the autoindexer polling directory. **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files** **mkdir test** **cp /tmp/goodbye.htm /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/test/**	goodbye.htm is copied to /var/system/autonomy/autonomy_db/KnowServer210/autoindexer/files/test/ on Autonomy machine.
13.	Copy goodbye.htm into the autoindexer polling directory. **telnet** <*web server machine*> **mkdir /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/test** **exit** **ftp** <*web server machine*> **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer/files/test** **lcd /tmp** **put goodbye.htm** **quit**	goodbye.htm is copied to /var/system/autonomy/autonomy_db/KnowServer210/autoindexer/files/test/ on web server machine.
14.	The autoindexer should automatically recognize the goodbye.htm file in the polling directory and begin the process to import into the DRE. Look at the autoindexer log file again to verify that hello.html was imported and indexed. **cd /var/*system*/autonomy/autonomy_db/KnowServer210/autoindexer** **tail KnowServer210autoindexer.log**	The autoindexer log should contain lines indicating that the IMPORT COMMAND and INDEXING COMMAND were successful.

#	User Action	Expected Observable Result
15.	Check that the DRE contains the document added by the autoindexer. Bring up the Knowledge Server configuration page in the web browser. **http://***<autonomy machine IP>***:***<data query port>***/qmethod=v**	The browser indicates that one document was added to the DRE. It will be of the form below: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 60000, Index port: 60001 2 documents 2 document sections 2 document slots 8 Terms Process ID: 24688 Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …

3.13.5. Test Scenario Expected Results

The test scenario expected results are that the following:

- Autoindexer recognizes content (Step 9)
- Autoindexer imports and indexes content into DRE (Step 10)
- Autoindexer recognizes content in subdirectory tree (Step 14)
- Autoindexer imports and indexes content in subdirectory tree (Step 15)

3.13.6. Test Scenario Acceptance Criteria

The response shown in the browser and log files matches with Expected Observable Result in Test Scenario procedure steps 9, 10, 14, and 15.

3.14. Scenario # 14 – Autonomy – DRE Query Validation (Local Host)

3.14.1. Test Scenario Description

This test validates the following:

1. That the Knowledge Server is configured properly to execute local queries on the Autonomy server

2. That the DRE accepts queries through a local web browser on the Autonomy Server and returns results to the local web browser.

3.14.2. Test Scenario Dependencies and Required Resources

Table 32 – Test Scenario # 14 – Autonomy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems			None
Personnel		✓	User
Test Data			None
Prior Testing	✓		Scenario # 13 - Internal Content Retrieval (File System)
Hardware		✓	Autonomy machine – SU35E2 Web server machine – SU35E3
Software		✓	Web browser, Autonomy Knowledge Server
Time to be Allotted			30 / 50 minutes

3.14.3. Test Scenario Detailed Design Description

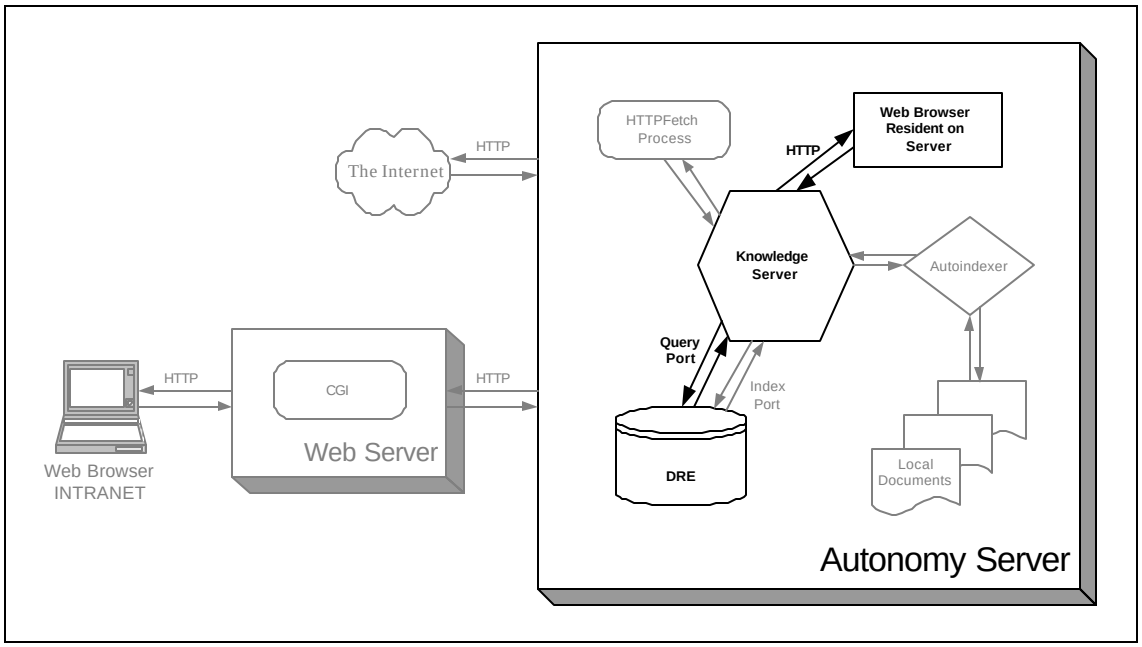

Figure 18 – Test Scenario # 14 – Autonomy – Detail Diagram

1. Using a Web browser resident on the Autonomy server, a user makes a HTTP query request to the Knowledge Server.

2. The Knowledge Server processes the HTTP query request and sends it to the DRE using the query port.

3. The DRE creates a HTTP query response, sends it to the Knowledge Server using the query port, and then to the client browser.

3.14.4. Test Scenario Inputs

Table 33 – Test Scenario # 14 – Autonomy – Machine Ports

Environment	Data DRE Query Port	Data DRE Index Port	Community DRE Query Port	Community DRE Index Port	HTTPFetch Service Port	HTTPFetch Spider Port
Dev	60000	60001	60002	60003	60005	80
Tst	30000	30001	30002	30003	30005	80
Stg	40000	40001	40002	40003	40005	80

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

system* =** Replace this keyword with the appropriate environment abbreviation (e.g. /var/system***/autonomy is /var/dev/autonomy).

Table 34 – Test Scenario # 14 – Autonomy – Procedure Steps

#	User Action	Expected Observable Result
1.	Check if Knowledge Server process is running. **netstat –a | grep** *<query port #>*	If nothing is returned, then Knowledge Server process is not running and continue to step 2. If something is returned, skip to step 3.
2.	Start Knowledge Server process **cd /var/*system*/autonomy/autonomy_db/KnowServer210** **./StartQuery.sh** (inside script press [ENTER] twice) **ps –ef | grep queryh**	Knowledge Server process is started. The KnowServer210queryh process is shown in the process list.
3.	Launch a web browser: **telnet su35e2** **cd /opt/netscape** export DISPLAY=<IP address of current machine>:0.0 **./netscape &**	Netscape web browser is displayed on the screen.
4.	Check if the KnowlegdeServer is running successfully. Enter the following into the web browser's URL to bring up the Knowledge Server status. **http://***<autonomy machine IP>***:***<data query port>***/qmethod=v**	The response from the browser shown below indicates that the DRE is running successfully: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 10000, Index port: 10001 0 documents 0 document sections 0 document slots 0 Terms Process ID: xxxxx Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …

#	User Action	Expected Observable Result
5.	Check if the DRE is ready for a query. Enter following into the web browser's URL to submit a query: http: / / <web server IP>:<web server port> / qmethod=q&querytext=hello	The response from the browser shown below indicates that the DRE is ready to execute a query: … Numhits=1 Doc_name=C:\Temp\hello.htm URL_title= This is a test document Doc_id=0 Doc_weight=100 links=HELLO dbase=0 summary= CHECKSUM= DREDOCTYPE= BIAS= QuickSummary= Hello World ***Original Query:qmethod=q&querytext=hello …

3.14.5. Test Scenario Expected Results

The expected results are as follows:

- The DRE runs successfully (Step 4).
- The DRE can execute a query (Step 5).

3.14.6. Test Scenario Acceptance Criteria

The response shown in the browser matches with Expected Observable Result in Test Scenario procedure steps 4 and 5.

3.15. Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration)

3.15.1. Test Scenario Description

This test validates that the Knowledge Server is configured properly to execute queries in a distributed configuration between a web server and Autonomy server. The web server makes a request via a Common Gateway Interface (CGI) program to the Autonomy server. The Autonomy server processes the DRE query and returns results to the CGI program. Then the CGI program displays the results on the web browser.

3.15.2. Test Scenario Dependencies and Required Resources

Table 35 – Test Scenario # 15 – Autonomy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems			None
Personnel		✓	User
Test Data			None
Prior Testing			None
Hardware		✓	Autonomy machine – SU35E2 Web server machine – SU35E3
Software		✓	Web browser, Autonomy Knowledge Server
Time to be Allotted			15 – 30 minutes

3.15.3. Test Scenario Detailed Design Description

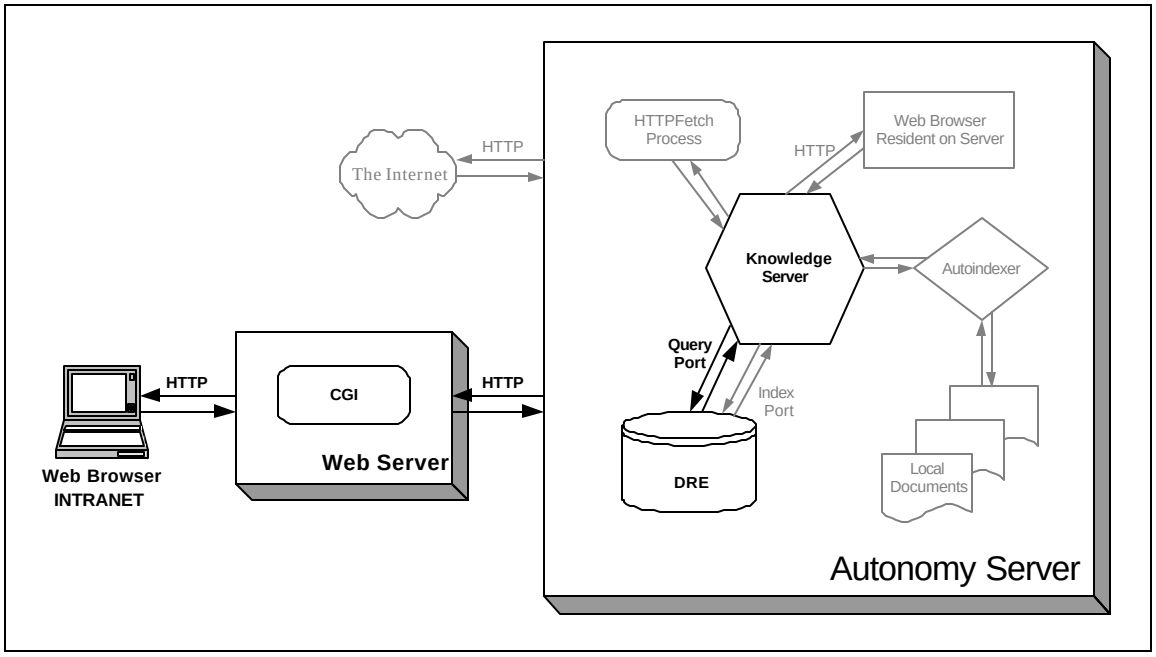

Figure 19 – Test Scenario # 15 – Autonomy – Detail Diagram

1. Using a Web browser on a client machine on the intranet, a user makes a query request.

2. The query request is processed by the Knowledge Server CGI on the Web server and sent via HTTP to the Knowledge Server on the Autonomy Server.

3. The Knowledge Server processes the HTTP query request and sends it to the DRE using the query port.

4. The DRE creates an HTTP query response and sends it to the Knowledge Server using the query port.

5. The Knowledge Server on the Autonomy Server sends the response to the Knowledge Server CGI on the Web Server and then to the user's Web browser.

3.15.4. Test Scenario Inputs

Table 36 – Test Scenario # 15 – Autonomy – Machine Ports

Environment	Data DRE Query Port	Data DRE Index Port	Community DRE Query Port	Community DRE Index Port	HTTPFetch Service Port	HTTPFetch Spider Port
Dev	60000	60001	60002	60003	60005	80
Tst	30000	30001	30002	30003	30005	80
Stg	40000	40001	40002	40003	40005	80

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

system = Replace this keyword with the appropriate environment abbreviation (e.g. /var/***system***/autonomy is /var/dev/autonomy).

Table 37 – Test Scenario # 15 – Autonomy – Procedure Steps

#	User Action	Expected Observable Result
1.	Check if Knowledge Server process is running. **netstat –a | grep** *<query port #>*	If nothing is returned, then Knowledge Server process is not running and continue to step 2. If something is returned, skip to step 3.
2.	Start Knowledge Server process **cd /var/*system*/autonomy/autonomy_db/KnowServer210** **./StartQuery.sh** (inside script press [ENTER] twice) **ps –ef | grep queryh**	Knowledge Server process is started. The KnowServer210queryh process is shown in the process list.
3.	Launch a web browser: **telnet su35e2** **cd /opt/netscape** export DISPLAY=<IP address of current machine>:0.0 **./netscape &**	Netscape web browser is displayed on the screen.
4.	Ensure that the KnowlegdeServer is running and ready to index content from the autoindexer. Enter this address into the web browser to bring up the Knowledge Server Configuration page. **http://***<autonomy machine IP>***:***<data query port>***/qmethod=v**	The response from the browser indicates information about the Knowledge Server running. It will be of the form below: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 60000, Index port: 60001 2 documents 2 document sections 2 document slots 8 Terms Process ID: 24688 Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …

#	User Action	Expected Observable Result
5.	Move to the Web Server machine and utilize the default Knowledge Server front-end web interface. Go to the web browser and enter: **http://***<web server IP>***:***<web server port>***/KnowServer210/**	A page should display with two frames. Left frame with autonomy logo and a "query " button. Right frame with a text box, "Query" button, and various check boxes.
6.	Type "**hello**" in the text box. Click on the "Query" button. (This searches the content added to the DRE in the autoindexer test. Be sure to run the autoindexer test first.)	The browser displays 1 search result. It should resemble the text below. Select Weight Title 79% This is a test document [Original Document][Plain Text] Hello World
7.	Click on the "This is a test document" link in the web browser page to bring up the document.	The browser displays the hello.htm page. **Hello World**

3.15.5. Test Scenario Expected Results

The expected results are as follows:

- Knowledge Server CGI comes up in browser (Step 5).
- Knowledge Server CGI on web server communicates with Autonomy server DRE and returns results (Step 6).
- Autoindexer recognizes content in subdirectory tree (Step 7).

3.15.6. Test Scenario Acceptance Criteria

The response shown in the browser matches with Expected Observable Result in Test procedure steps 5, 6, and 7.

3.16. Scenario # 16 – Autonomy – External Content Retrieval (Internet)

3.16.1. Test Scenario Description

The Autonomy HTTPFetch uses a multi-socked, parallel approach to retrieving content from remote web sites such as www.ed.gov. The HTTPFetch consists of three processes: ***spidering, importing and indexing.*** The documents are gathered by HTTP requests in the spidering process, and then are converted into the DRE's indexing format - idx files- in the importing process. Finally the idx files are indexed into the DRE during the indexing process. The failure of any of these three processes results in no content being indexed into the DRE. This test verifies that each of these processes run successfully.

3.16.2. Test Scenario Dependencies and Required Resources

Table 38 – Test Scenario # 16 – Autonomy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems			None
Personnel		✓	User
Test Data			None
Prior Testing			None
Hardware		✓	Autonomy machine – SU35E2 Web server machine – SU35E3
Software		✓	Web browser, Autonomy Knowledge Server
Time to be Allotted			Several hours depending on the size of a site

3.16.3. Test Scenario Detailed Design Description

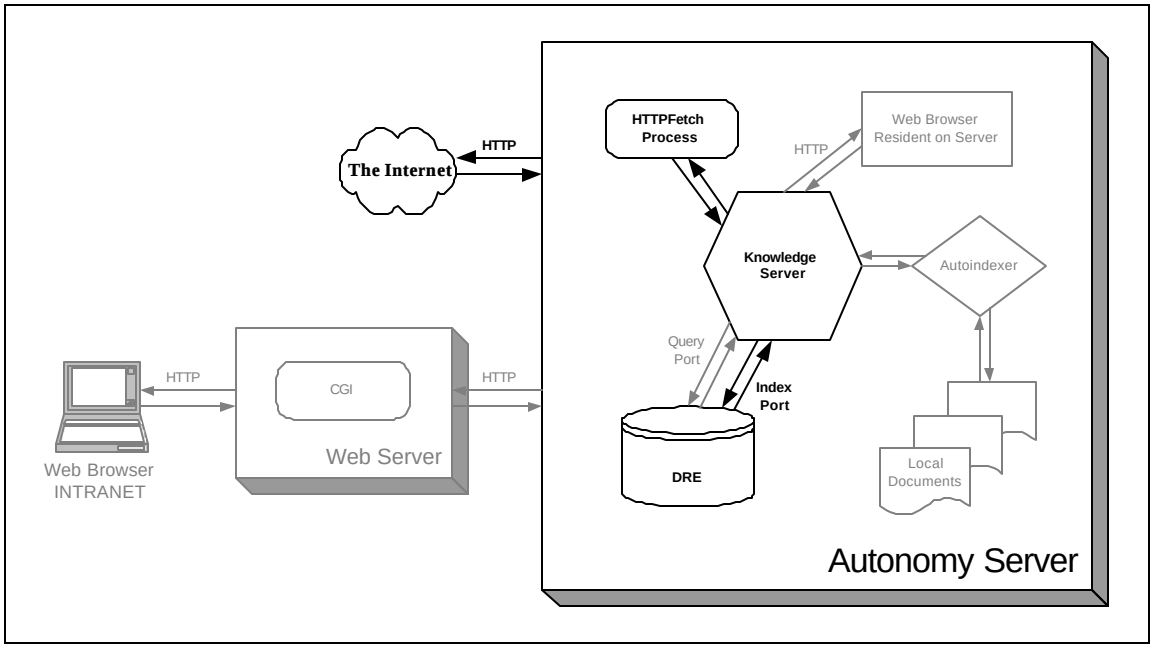

Figure 20 – Test Scenario # 16 – Autonomy – Detail Diagram

1. The HTTPFetch periodically spiders Web sites specified in its configuration file.
2. HTTPFetch downloads content from these Web sites and indexes the documents.
3. The indexed documents are then imported to the Knowledge Server.
4. The Knowledge Server then imports the indexed documents into the DRE using the index port.
5. The DRE then sends a success or failure message through the Knowledge Server to the HTTPFetch, on whether or not the documents where successfully indexed into the DRE.

3.16.4. Test Scenario Inputs

Table 39 – Test Scenario # 16 – Autonomy – Machine Ports

Environment	Data DRE Query Port	Data DRE Index Port	Community DRE Query Port	Community DRE Index Port	HTTPFetch Service Port	HTTPFetch Spider Port
Dev	60000	60001	60002	60003	60005	80
Tst	30000	30001	30002	30003	30005	80
Stg	40000	40001	40002	40003	40005	80

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

system* =** Replace this keyword with the appropriate environment abbreviation (e.g. /var/system***/autonomy is /var/dev/autonomy).

Table 40 – Test Scenario # 16 – Autonomy – Procedure Steps

#	User Action	Expected Observable Result
1.	Configure HTTPFetch210 spider by editing HTTPFetch210.cfg **cd /var/*system*/autonomy/autonomy_db** **/HTTPFetch210** **vi HTTPFetch210.cfg**	Edit/add these lines in the [SPIDER] section [SPIDER] … NUMBER=2 0=AUTONOMY 1=TEST [AUTONOMY] URL=http://www.autonomy.com [TEST] URL=http://4.20.14.158/ DATABASE=Archive LOG FILE='TEST.log DATECHECK=0 BEFOREDATE=10000 AFTERDATE=-10000 …
2.	Start HTTPFetch210 **cd /var/*system* /autononmy/autonomy_db** **/HTTPFetch210** **./startfetch.sh** Check to see if the fetch process successfully started **ps –ef | grep HTTPF**	If HTTPFetch process is started, the HTTPFetch210 process is shown in the process list.

#	User Action	Expected Observable Result
3.	Check the spidering process by monitoring the spider log file. **cd /var/*system*/autononmy/autonomy_db** **/HTTPFetch210** **tail –f TEST.log**	Portion of TEST.log file is shown as follows: … TEST : Getting link http://4.20.14.158/privacy.shtml TEST : Links:37 (New Valid Links:0) TEST : Saved page http://4.20.14.158/overview.shtml TEST : Current flow rate: 4.25kbps (of 7.52kbps) TEST : Getting link http://4.20.14.158/links.shtml TEST : Links:30 (New Valid Links:0) TEST : Saved page http://4.20.14.158/privacy.shtml TEST : Current flow rate: 5.27kbps (of 10.34kbps) TEST : Getting link http://4.20.14.158/glossary.shtml TEST : Links:34 (New Valid Links:0) TEST : Saved page http://4.20.14.158/links.shtml TEST : Current flow rate: 5.97kbps (of 17.20kbps) …
4.	Check the importing and indexing processes by monitoring the HTTPFetch210spider.log **cd /var/*system*/autononmy/autonomy_db** **/HTTPFetch210** **tail –f HTTPFetchspider.log**	… Import: Importing TEST to /var/tst/autonomy/autonomy_db/HTTPFetch210/TEST.idx... Import: Success Index: Indexing '/var/tst/autonomy/autonomy_db/HTTPFetch210/TEST.idx' as '/var/tst/autonomy/autonomy_db/HTTPFetch210/TEST.idx' -> DRE on '127.0.0.1:30001'... (over socket): Success …

#	User Action	Expected Observable Result
5.	Check the DRE status to verify if the fetching process is successfully completed. Enter this address into the web browser to bring up the Knowledge Server status page. **http://***<autonomy machine IP>***:***<data query port>***/qmethod=v**	The response from the browser shown below indicates that the fetching process is completed successfully: Autonomy DRE server version 3.1.0 Build 2.45 Query port: 30000, Index port: 30001 10 documents 11 document sections 10 document slots 120 Terms Process ID: xxx Directory: /autonomy/tst/autonomy/autonomy_db/KnowServer210/querydre 0 DateFix(s): 2 Database(s): Name Expiry Expiry Max Date Security Documents Documents Hours Action Hits Range Sections 0:Aut Yes 0 0 onomy 1:Test Yes 10 11 …
6.	Query the DRE to verify if the content indexed by the fetching process is retrievable. Enter this address into the web browser to bring up the Knowledge Server GUI front end **http://***<web server>***:***<web server port>***/KnowServer210/**	A page with two frames. Left page has Autonomy Logo and a "query" button; right frame has a text box and various check boxes.
7.	Prepare a query by typing in "loan", then click "Query" button, and then click "OK".	Numerous query results are shown in the browser with selection checkbox, weight indication, title and summary for each document.
8.	Open a query result : Click on a link title on the query results list	The document is opened in the upper portion of the browser.

3.16.5. Test Scenario Expected Results

The expected results are as follows:

- Fetching process is going successfully (Step 3).
- Importing and indexing processes are completed successfully (Step 4).

- The DRE status page shows new content indexed (Step 5).
- The new content in the DRE is retrievable (Step 6, 7 and 8).

3.16.6. Test Scenario Acceptance Criteria

The response shown in the browser and log files matches with Expected Observable Result in Test Scenario procedure steps 3, 4, 5, 7 and 8.

3.17. Scenario # 17 – Autonomy – External Content Scheduled Retrieval (Internet)

3.17.1. Test Scenario Description

The schedule for the HTTPFetch process has two main components, a starting time and an interval between each execution. This test validates both HTTPFetch schedule components by:

1. Having the HTTPFetch process start automatically at a specified time.

2. Ensuring that the HTTPFetch process repeats at a specified interval.

3.17.2. Test Scenario Dependencies and Required Resources

Table 41 – Test Scenario # 17 – Autonomy – Dependencies and Resource Requirements

D = Dependency; R = Required Resource

Category	D	R	Description
Other Systems			None
Personnel		✓	User
Test Data			None
Prior Testing			None
Hardware		✓	Autonomy machine – SU35E2 Web server machine – SU35E3
Software		✓	Web browser, Autonomy Knowledge Server
Time to be Allotted			10 minutes

3.17.3. Test Scenario Detailed Design Description

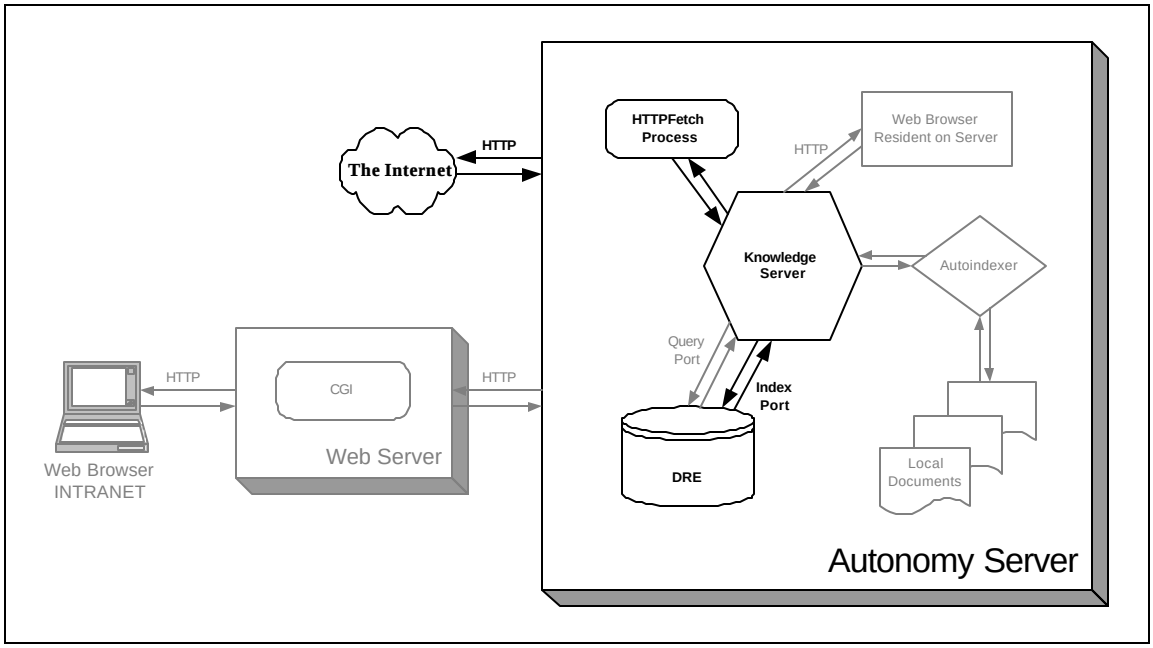

Figure 21 – Test Scenario # 17 – Autonomy – Detail Diagram

The HTTPFetch configuration accepts four parameters that handle the scheduling:

- *SpiderRepeatInterval*: The time between spider cycles in hours or days.
- *SpiderRepeatSecs*: The time before the spiders should repeat in seconds. SpiderRepeatInterval overrides it.
- *SpiderCycles*: The number of times a spider should attempt to index a site. –1 means infinite of times of repeats.
- *SpiderStartTime*: The clock time to begin the spider cycle in twenty-four hour time such as 23:30.

This test is aimed at testing HTTPFetch process with two-scheduled events, starting up the spider and repeating the spidering process. The main objectives are to demonstrate the spidering process starts up automatically at the time specified by *SpiderStartTime,* and then is repeated following the time interval specified by *SpiderRepeatSecs.*

3.17.4. Test Scenario Inputs

Table 42 – Test Scenario # 17 – Autonomy – Machine Ports

Environment	Data DRE Query Port	Data DRE Index Port	Community DRE Query Port	Community DRE Index Port	HTTPFetch Service Port	HTTPFetch Spider Port
dev	60000	60001	60002	60003	60005	80
tst	30000	30001	30002	30003	30005	80
stg	40000	40001	40002	40003	40005	80

Test Procedure Steps

Conventions: For text to be input, **Bold** is a literal to be entered exactly as shown, *Italics* is a variable to be replaced with the value provided or described.

system* =** Replace this keyword with the appropriate environment abbreviation (e.g. /var/system***/autonomy is /var/dev/autonomy).

Table 43 – Test Scenario # 17 – Autonomy – Procedure Steps

#	User Action	Expected Observable Result
1.	Check if Knowledge Server process is running. **Netstat –a | grep** *<query port #>*	If nothing is returned, then Knowledge Server process is not running and continue to step 2. If something is returned, skip to step 3.
2.	Start Knowledge Server process **cd /var/*system*/autonomy/autonomy_db/KnowServer210** **./StartQuery.sh** (inside script press [ENTER] twice) **ps –ef | grep queryh**	Knowledge Server process is started. The KnowServer210queryh process is shown in the process list.
3.	Launch a web browser: **Telnet su35e2** **cd /opt/netscape** Export DISPLAY=<IP address of current machine>:0.0 **./netscape &**	Netscape web browser is displayed on the screen.

#	User Action	Expected Observable Result
4.	Ensure that the KnowlegdeServer is running and ready to index content from the autoindexer. Enter this address into the web browser to bring up the Knowledge Server Configuration page. **Http://***<autonomy machine IP>***:***<data query port>***/qmethod=v**	The response from the browser indicates information about the Knowledge Server running. It will be of the form below: Autonomy DRE Server version 3.1.0 Build 2.45 Query port: 60000, Index port: 60001 2 documents 2 document sections 2 document slots 8 Terms Process ID: 24688 Directory: /var/autonomy/autonomy_db/KnowServer210/querydre …
5.	Move to the Web Server machine and utilize the default Knowledge Server front-end web interface. Go to the web browser and enter: **Http://***<web server IP>***:***<web server port>***/KnowServer210/**	A page should display with two frames. Left frame with autonomy logo and a "query " button. Right frame with a text box, "Query" button, and various check boxes.
6.	Type "**hello**" in the text box. Click on the "Query" button. (This searches the content added to the DRE in the autoindexer test. Be sure to run the autoindexer test first.)	The browser displays 1 search result. It should resemble the text below. Select Weight Title 79% This is a test document [Original Document][Plain Text] Hello World
7.	Click on the "This is a test document" link in the web browser page to bring up the document.	The browser displays the hello.htm page. **Hello World**

3.17.5. Test Scenario Expected Results

The test scenario expected results are that the following:

- Knowledge Server CGI comes up in browser (Step 5).
- Knowledge Server CGI on web server communicates with Autonomy server DRE and returns results (Step 6).
- Autoindexer recognizes content in subdirectory tree (Step 7).

3.17.6. Test Scenario Acceptance Criteria

The response shown in the browser matches with Expected Observable Result in Test procedure steps 5, 6, and 7.

3.18. Autonomy – Fail Over Testing Report

3.18.1. Fail Over Test Description

Introduction

The Autonomy service in the production environment is configured in a high-availability architecture, where the DRE is accessed through a Network Dispatcher managed Autonomy virtual cluster. The Autonomy virtual cluster contains two Autonomy machines with synchronized DREs. There are two important parts of this high-availability architecture. The first is the use of Network Dispatcher to manage the Autonomy service and detection of an Autonomy machine failure. The second is keeping data contained in the two Autonomy machine DREs synchronized.

Autonomy is installed on SU35E7 and SU35E15, designated the primary and the secondary Autonomy machines, respectively. Both machines contain a DRE with synchronized content. The primary machine has the additional function of fetching external content from Internet into both the primary and secondary machine DREs.

Test Description

To validate the high-availability of the Autonomy service, a series of fail over tests were designed and conducted. These tests are categorized as follows:

1) Autonomy Virtual Cluster Testing

2) DRE Content Synchronization Testing

The Autonomy Virtual Cluster Testing ensures that Autonomy queries function properly even if one Autonomy machine is not running and the load is balanced between Autonomy Machines.

The DRE Content Synchronization Testing ensures that the two Autonomy machines maintain synchronized content. This is necessary in the event of one machine failing or in the load balancing scheme used between the two Autonomy machines.

Conclusion

The Autonomy service on the production environment is configured and tested as a high-availability architecture. The Autonomy Virtual Cluster Testing and DRE Content Synchronization Testing cover three areas:

- Autonomy product installation on the primary and secondary machine
- Autonomy Virtual Cluster reliability and load-balancing

- DRE content synchronization on both primary and secondary machines.

All tests were performed.

3.18.2. Autonomy Virtual Cluster Testing

The Fail Over test of the Autonomy Virtual Cluster utilized the following Test Scenarios with varying machine combinations. For more details, refer to the sections corresponding to the following scenarios:

- Scenario #13 – Autonomy – Internal Content Retrieval (File System)
- Scenario #14 – Autonomy – DRE Query Validation (Local Configuration)
- Scenario #15 – Autonomy – DRE Query Validation (Distributed Configuration)
- Scenario #16 – Autonomy – External Content Retrieval (Internet)

Test Scenarios

- Scenario A – Primary Autonomy Machine has all functionality with individual configuration
- Scenario B – Secondary Autonomy Machine has all functionality with individual configuration
- Scenario C – Network Dispatcher balances load on Autonomy cluster with both machines running
- Scenario D – Queries execute successfully using Autonomy cluster with both machines running
- Scenario E – Queries execute successfully using Autonomy cluster with the primary machine down
- Scenario F – Queries can execute successfully using Autonomy cluster with the secondary machine down
- Scenario G – Network Dispatcher balances load on Autonomy cluster when primary machine is restored and secondary machine is already running
- Scenario H – Network Dispatcher balances load on Autonomy cluster when the secondary machine is restored and the primary machine already running

Table 44 – Autonomy Fail Over – Virtual Cluster – Scenario A

Scenario A - Primary Autonomy Machine has all functionality with individual configuration	
Purpose	This scenario verifies that the primary machine has all functionality when configured individually.
Setup	Configure primary Autonomy Machine with the primary machine IP address.
Procedure & Expected Results	Complete the following Autonomy test scenarios using the primary machine IP address: • Scenario # 13 – Autonomy – Internal Content Retrieval (File System) • Scenario # 14 – Autonomy – DRE Query Validation (Local Configuration) • Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration) • Scenario # 16 – Autonomy – External Content Retrieval (Internet)
Actual Results	Test completed with results matching the "Test Scenario Expected Results" sections from scenarios listed above.

Table 45 – Autonomy Fail Over – Virtual Cluster – Scenario B

Scenario B - Secondary Autonomy Machine has all functionality with individual configuration	
Purpose	This scenario verifies that the secondary machine has all functionality when configured individually.
Setup	Configure secondary Autonomy Machine with the secondary machine IP address.
Procedure & Expected Results	Complete the following Autonomy test scenarios using the secondary machine IP address: • Scenario # 13 – Autonomy – Internal Content Retrieval (File System) • Scenario # 14 – Autonomy – DRE Query Validation (Local Configuration) • Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration) • Scenario # 16 – Autonomy – External Content Retrieval (Internet)
Actual Results	Test completed with results matching the "Test Scenario Expected Results" sections from scenarios listed above.

Table 46 – Autonomy Fail Over – Virtual Cluster – Scenario C

Scenario C - Network Dispatcher balances load on Autonomy cluster with both machines running	
Purpose	This scenario verifies that the Network Dispatcher balances the load between the Autonomy machines.
Setup	• Configure Autonomy Cluster on Network Dispatcher for load balancing between Autonomy machines. • Autonomy DRE running on both machines.
Procedure	1. Identify process ID for DRE on primary machine using configuration page:

	http://<*IP address of primary machine*>:10000/qmethod=v 2. Identify process ID for DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v 3. Load Autonomy DRE configuration page using cluster IP address: http://<*cluster IP address*>:10000/qmethod=v 4. Record the process ID shown on this page, it will identify which machine that Network Dispatcher directed that query. 5. Hit the "Reload" button on the browser several times. After each time, record the process ID shown on the page.
Expected Results	The process ID shown on the configuration page should alternate between Autonomy machines. This verifies that Network Dispatcher is balancing query loads between the Autonomy machines.
Actual Results	The different process IDs shown in the web browser verify that load balancing is occurring between the Autonomy machines.

Table 47 – Autonomy Fail Over – Virtual Cluster – Scenario D

Scenario D – Queries execute successfully using Autonomy cluster with both machines running	
Purpose	This scenario verifies that queries execute successfully using the Autonomy cluster.
Setup	• Web server is configured to query using the Autonomy cluster IP address. • Autonomy DRE running on both machines.
Procedure & Expected Results	Complete the following Autonomy test scenarios: • Scenario # 14 – Autonomy – DRE Query Validation (Local Configuration) • Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration)
Actual Results	Test completed with results matching the "Test Scenario Expected Results" sections from scenarios listed above.

Table 48 – Autonomy Fail Over – Virtual Cluster – Scenario E

Scenario E - Queries execute successfully using Autonomy cluster with the primary machine down	
Purpose	This scenario verifies that queries execute successfully using the Autonomy cluster IP when the primary machine is down. This verifies that Network Dispatcher recognizes that the primary machine is down and directs queries only to the secondary machine.
Setup	• Web server CGI is configured to query using the Autonomy cluster IP address. • Autonomy DRE on primary machine is not running. • Autonomy DRE on secondary machine is running.
Procedure & Expected Results	Complete the following Autonomy test scenarios: • Scenario # 14 – Autonomy – DRE Query Validation (Local Configuration)

	• Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration)
Actual Results	Test completed with results matching the "Test Scenario Expected Results" sections from scenarios listed above.

Table 49 – Autonomy Fail Over – Virtual Cluster – Scenario F

Scenario F - Queries execute successfully using Autonomy cluster with the secondary machine down	
Purpose	This scenario verifies that queries execute successfully using the Autonomy cluster IP when the secondary machine is down. This verifies that Network Dispatcher recognizes that the secondary machine is down and directs queries only to the primary machine.
Setup	• Web server CGI is configured to query using the Autonomy cluster IP address. • Autonomy DRE on primary machine is running. • Autonomy DRE on secondary machine is not running.
Procedure & Expected Results	Complete the following Autonomy test scenarios: • Scenario # 14 – Autonomy – DRE Query Validation (Local Configuration) • Scenario # 15 – Autonomy – DRE Query Validation (Distributed Configuration)
Actual Results	Test completed with results matching the "Test Scenario Expected Results" sections from scenarios listed above.

Table 50 – Autonomy Fail Over – Virtual Cluster – Scenario G

Scenario G - Network Dispatcher balances load on Autonomy cluster when primary machine is restored and secondary machine is already running	
Purpose	This verifies that Network Dispatcher recognizes when the primary machine is restored and enables load balancing by directing queries to both machines.
Setup	• Autonomy DRE on the primary machine is not running. • Autonomy DRE on secondary machine is running.
Procedure	1. Identify process ID for DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v 2. Load Autonomy DRE configuration page using cluster IP address: http://<*cluster IP address*>:10000/qmethod=v 3. Record the process ID shown on this page, it should match the process ID from secondary machine. 4. Start the Autonomy DRE on primary machine. 5. Wait 1 minute. 6. Identify process ID for DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 7. Load Autonomy DRE configuration page using cluster IP address:

	http://<*IP address of secondary machine*>:10000/qmethod=v 8. Hit the "Reload" button on the browser several times. After each time, record the process ID shown on the page.
Expected Results	The process ID shown on the configuration page should alternate between Autonomy machines. This verifies that Network Dispatcher is recognizes when the primary machine is restored and load balances queries between Autonomy machines.
Actual Results	The different process IDs shown in the web browser verifies that there is load balancing between the Autonomy machines when the primary machine is restored.

Table 51 – Autonomy Fail Over – Virtual Cluster – Scenario H

Scenario H - Network Dispatcher balances load on Autonomy cluster when the secondary machine is restored and the primary machine already running	
Purpose	This verifies that Network Dispatcher recognizes that the secondary machine is restored and enables load balancing by directing queries to both machines.
Setup	• Autonomy DRE on secondary machine is not running. • Autonomy DRE on primary machine is running
Procedure	1. Identify process ID for DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 2. Load Autonomy DRE configuration page using cluster IP address: http://<*cluster IP address*>:10000/qmethod=v 3. Record the process ID shown on this page, it should match the process ID from the primary machine. 4. Start the Autonomy DRE on the secondary machine. 5. Wait 1 minute. 6. Identify process ID for DRE on the secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v 7. Load Autonomy DRE configuration page using cluster IP address http://<*cluster IP address*>:10000/qmethod=v 8. Hit the "Reload" button on the browser several times. After each time, record the process ID shown on the page.
Expected Results	The process ID shown on the configuration page should alternate between Autonomy machines. This verifies that Network Dispatcher is recognizes when the secondary machine is restored and load balances queries between Autonomy machines.
Actual Results	The different process IDs shown in the web browser verifies that there is load balancing between the Autonomy machines when the secondary machine is restored.

3.18.3. DRE Content Synchronization Test

This section has two groups of test scenarios to assure that the content of the DRE is always synchronized from the fetching and autoindexer processes.

- Scenarios I to M are related to the HTTPFetch process
  - Scenario I – HTTPFetch: DREs are synchronized under normal conditions
  - Scenario J – HTTPFetch: Primary Machine DRE is indexed with secondary DRE not running
  - Scenario K – HTTPFetch: DREs are synchronized under abnormal conditions (primary machine DRE has content, secondary machine DRE has no content)
  - Scenario L – HTTPFetch: Secondary machine DRE is indexed with primary machine DRE not running
  - Scenario M – HTTPFetch: DREs are synchronized under abnormal conditions (secondary machine DRE has content, primary machine DRE has no content)
- Scenarios N to R are related to the autoindexer process
  - Scenario N – Autoindexer: DREs are synchronized under normal conditions
  - Scenario O – Autoindexer: Primary machine DRE is indexed with secondary machine DRE not running
  - Scenario P – Autoindexer: DREs are synchronized under abnormal conditions (primary machine DRE has content, secondary machine DRE does not)
  - Scenario Q – Autoindexer: Secondary machine DRE is indexed with primary machine DRE not running
  - Scenario R – Autoindexer: DREs are synchronized under abnormal conditions (secondary machine DRE has content, primary machine DRE does not)

Table 52 – Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario I

Scenario I - HTTPFetch: DREs are synchronized under normal conditions	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary DREs with HTTPFetch under normal conditions.
Setup	The DRE on the primary and secondary machines is running and contains no content.
Procedure	1. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 2. Identify number of documents in DRE on primary machine using configuration page:

	http://<*IP address of primary machine*>:10000/qmethod=v 3. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the primary and secondary machines has the same number of documents.
Actual Results	The test was completed and the results matched the expected results listed above.

Table 53 – Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario J

Scenario J - HTTPFetch: Primary Machine DRE is indexed with secondary DRE not running	
Purpose	This scenario validates that the primary machine DRE is indexed with HTTPFetch even if the secondary machine DRE is not running.
Setup	• The DRE on the primary machine is running and contains no content. • The DRE on the secondary machine is not running
Procedure	1. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 2. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v
Expected Results	The DRE on the primary machine contains documents from the HTTPFetch external content retrieval.
Actual Results	The test was completed and the results matched the expected results listed above.

Table 54 – Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario K

Scenario K - HTTPFetch: DREs are synchronized under abnormal conditions (primary machine DRE has content, secondary machine DRE has no content)	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary machine DREs with HTTPFetch under abnormal conditions, that is the primary machine DRE has already had content, while the secondary machine DRE has no content.
Setup	• Complete Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario J (this puts content into the primary DRE, leaves the secondary DRE with no content). • The primary and secondary machine DREs are running.
Procedure	1. Run reset_httpfetch.sh to reset HTTPFetch status. 2. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 3. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 4. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine is indexed with new content. The primary and secondary DREs have the same number of documents.

Actual Results	The test was completed with results matching the expected results listed above.

Table 55 – Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario L

Scenario L - HTTPFetch: Secondary machine DRE is indexed with primary machine DRE not running	
Purpose	This scenario validates that the secondary machine DRE is indexed with HTTPFetch even if the primary machine DRE is not running.
Setup	• The DRE on the secondary machine is running and contains no content. • The DRE on the primary machine is not running.
Procedure	1. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 2. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine contains documents from the HTTPFetch external content retrieval.
Actual Results	The test was completed and the results match the expected results listed above.

Table 56 – Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario M

Scenario M - HTTPFetch: DREs are synchronized under abnormal conditions (secondary machine DRE has content, primary machine DRE has no content)	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary DREs with HTTPFetch under abnormal conditions, that is the secondary machine DRE has already has content, while the primary machine DRE has no content.
Setup	• Complete Autonomy Fail Over – DRE Content Synchronization – HTTPFetch – Scenario L (this puts content into the secondary DRE, leaves the primary DRE with no content). • The primary and secondary machine DREs are running.
Procedure	1. Run reset_httpfetch.sh to reset HTTPFetch status. 2. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 3. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 4. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine is indexed with new content. The DREs on the primary and secondary machine have the same number of documents.
Actual Results	The test was completed with results matching the expected results listed above.

Table 57 – Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario N

Scenario N - Autoindexer: DREs are synchronized under normal conditions	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary machine DREs with Autoindexer under normal conditions.
Setup	The DRE on the primary and secondary machines is running and contains no content.
Procedure	1. Complete Scenario # 13 – Autonomy – Internal Content Retrieval (File System). 2. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 3. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the primary and secondary machines have same of the number of documents.
Actual Results	The test was completed and the DRE on the primary and secondary machines contained the same number of documents.

Table 58 – Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario O

Scenario O - Autoindexer: Primary machine DRE is indexed with secondary machine DRE not running	
Purpose	This scenario validates that the primary machine DRE is indexed with autoindexer even if the secondary machine DRE is not running.
Setup	• The DRE on the primary machine is running and contains no content. • The DRE on the secondary machine is not running.
Procedure	1. Complete Scenario # 13 – Autonomy – Internal Content Retrieval (File System). 2. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v
Expected Results	The DRE on the primary machine contains documents.
Actual Results	The test was completed with results matching the expected results listed above.

Table 59 – Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario P

Scenario P - Autoindexer : DREs are synchronized under abnormal conditions (primary machine DRE has content, secondary machine DRE does not)	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary machine DREs with autoindexer under abnormal conditions, that is the primary machine DRE has already had content, while the secondary machine DRE has no content.
Setup	• Complete Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario O (this puts content into the primary machine DRE, leaves the secondary machine DRE with no content). • The primary and secondary machine DREs are running.

Procedure	1. Run reset_autoindexer.sh to reset the Autoindexer status. 2. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 3. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v 4. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine is indexed with new content. The primary and secondary DREs contain the same number of documents.
Actual Results	The test was completed with results matching the expected results listed above.

Table 60 – Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario Q

Scenario Q - Autoindexer: Secondary machine DRE is indexed with primary machine DRE not running	
Purpose	This scenario validates that the secondary machine DRE is indexed with autoindexer even if the primary machine DRE is not running.
Setup	• The DRE on the secondary machine is running and contains no content. • The DRE on the primary machine is not running.
Procedure	1. Complete Scenario # 13 – Autonomy – Internal Content Retrieval (File System). 2. Identify number of documents in DRE on the secondary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine contains documents.
Actual Results	The test was completed with results matching the expected results listed above.

Table 61 – Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario R

Scenario R - Autoindexer: DREs are synchronized under abnormal conditions (secondary machine DRE has content, primary machine DRE does not)	
Purpose	This scenario validates the synchronization of the contents of the primary and secondary machine DREs with autoindexer under abnormal conditions, that is the secondary machine DRE has already had content, while the primary machine DRE has no content.
Setup	• Complete Autonomy Fail Over – DRE Content Synchronization – Autoindexer – Scenario Q (this puts content into the secondary DRE, leaves the primary DRE with no content). • The primary and secondary machine DRE are running.
Procedure	1. Run reset_autoindexer.sh to reset the Autoindexer status. 2. Complete Scenario # 16 – Autonomy – External Content Retrieval (Internet). 3. Identify number of documents in DRE on primary machine using configuration page: http://<*IP address of primary machine*>:10000/qmethod=v

	4. Identify number of documents in DRE on secondary machine using configuration page: http://<*IP address of secondary machine*>:10000/qmethod=v
Expected Results	The DRE on the secondary machine is indexed with new content. The DREs on the primary and secondary machines contain the same number of documents.
Actual Results	The test was completed with results matching the expected results listed above.

4 ITA Test Scenarios – Post Release 1

This section contains the test scenarios that will be designed and executed for the next release of the ITA. Each anticipated scenario is listed along with the description of the test objective(s). There are two subsections - Informatica and IHS – and an overview section precedes the IHS listing.

4.1. Informatica Test Scenarios

Table 62 – ITA Test Scenarios – Post Release 1 – Informatica

#	Scenario Name	Scenario Description
18	Informatica - Extract Data from a File	This test will verify that the Informatica server can source (extract) data from a legacy DB2 mainframe. As a prerequisite the DB2 Connect drivers must be installed and configured properly on the server. Once the DB2 Connect drivers are installed and configured, then a sample file will be used as a source. A simple mapping will be developed to extract this data, and the output (load) will be a flat file that is created and written on a directory local to the server.
19	Informatica – Extract Data from a Database	This test will verify that the Informatica server can source (extract) data from a legacy database. As a prerequisite the DB2 Connect drivers must be installed and configured properly on the server. Once the DB2 Connect drivers are installed and configured, then a direct connection to the database will be possible. A simple mapping will be developed to extract this data, and the output (load) will be a flat file that is created and written on a directory local to the server.
20	Informatica – File Transfer (FTP) Data to the Informatica Server	This test will verify that files can be sent via File Transfer Protocol (FTP) to the Informatica Server. As a basis for executing this test the Central Data System (CDS) Retirement application will be used to reduce the amount of custom development. The INSTfile will be sent from the IBM mainframe and the DDR and Title IV Wide Area Network (TIVWAN) files will be FTP'd from the VAX to the server.
21	Informatica – Data Transformation and Loading	This test will verify that processing (transformations) occur on the server and are then loaded successfully into the Oracle schema. Once development for CDS retirement is complete, the repository will be imported into the Vicdev tablespace at the VDC. This will replicate the Informatica development.

4.2. IHS – Reliability, Availability, Scalability (RAS)

4.2.1. Overview

When determining if a system architecture meets a customer's Reliability, Scalability, and Availability (RAS) needs, the architecture needs to be investigated for possible points of failure. With the points of failure identified, redundancy and duplication can be designed

and built into the architecture that enable the continued servicing of requests during a partial outage.

To ensure reliability, availability, and scalability in the architecture, the physical topology of the architecture will consist of:

- A layer of machines running Network Dispatcher (ND)
- A layer of machines running IHS
- A layer of machines running WAS admin servers and WAS application servers
- A database server supporting the WAS admin repository

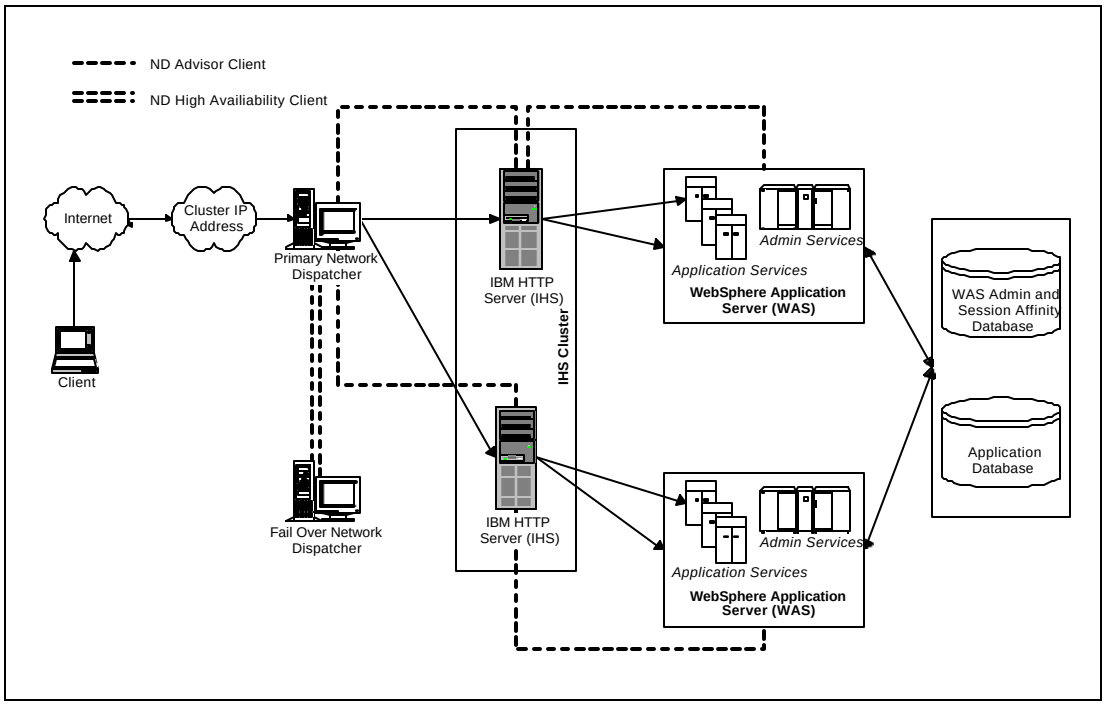

Figure 22 – IHS RAS Overview – Detail Diagram

Network Dispatcher (ND)

Network Dispatcher (ND) is a high availability and load balancing software server. ND handles all client requests through the implementation of clusters, a set of locally managed machines to be administered as a single logical unit. In the above configuration, a cluster has been created on two IBM HTTP Servers (IHS).

Network Dispatcher will monitor the cluster, load balance the client requests to the clustered IHS servers, and will ensure session persistence through the sticky bit configuration. With this configuration, once a session is established on an IHS server, ND will return that session

to the IHS server where the session was established. If an IHS server fails, ND will detect the outage and will not route any more requests to the failed IHS server. Once the IHS server has been recovered, ND will detect the server and begin routing requests to the server.

Network Dispatcher has also been configured in high availability mode. One ND server is configured as the primary and the other ND server is configured as the fail over. If the primary ND fails, the fail over ND server will automatically take over routing client requests.

IBM Http Server (IHS)

IHS is a web server that processes html requests for information. In the above configuration, Network Dispatcher determines which of the clustered IHS servers should receive a client request by evaluating the current Network traffic and the sticky bit configuration. When the request is routed to the IHS server, the IHS server will determine if an application server, WebSphere Application Server, is needed to serve the request.

If the services of an application server are needed, the WebSphere OSE plug-in on the IHS server will submit the request to an application service for server side processing. The OSE plug-in will load balance the WAS application services and ensure that a client request is feed to a functional application service.

WebSphere Application Servers (WAS)

WebSphere Application Server is an application environment that supports dynamic web content generation. In the above configuration, a WAS server is dedicated to each IHS server to handle server side processing. This processing may include Java Script processing, Java Servlet processing, or Java Enterprise Bean connections.

WebSphere Application servers can be configured to handle application database connections, ensure session persistence, and provide application security. Each WebSphere Application Server has one admin service and one or more application services. The admin and application services are monitored by a nanny process that will restart these processes in case of failure.

To ensure scalability, a properly configured WAS application service, a master application service, can be cloned so that an IHS machine can feed two or more identical application services. This is referred to as vertical scaling and is especially helpful in situations where a WAS server has excess CPU cycles and can handle more then one application service. The cloned service also stands as an additional backup in case the master application service fails. A cloned service can also be configured on a separate WAS server to allow additional processing power. This is referred to as horizontal scaling.

Database Server

In the above configuration, the WebSphere Application Servers require a database for the administration repository and session persistence data. A database server will also be

required for web applications that store and retrieve user information within a database. The database server needs to be fault tolerant since many aspects of the architecture and the application depends on the database server being available.

4.3. IHS Test Scenarios

Test scenarios have been identified that stress the reliability, availability, and scalability of the architecture. The following sections detail these scenarios and the expected system actions that will compensate for the partial failures.

Table 63 – ITA Test Scenarios – Post Release 1 – IHS

#	Scenario Name	Scenario Description
22	RAS – WebSphere Session Persistence	This test will verify that WAS correctly utilizes session persistence to enable multiple application clones (possibly on different machines) to share the same HTTP session information. WAS automatically takes cares of the details of managing this state. WAS uses a database (such as Oracle or DB2) for this purpose.
23	RAS – Network Dispatcher (ND) Load Balancing to IHS	This test will verify (1) that ND load balances different user requests across the available IHS servers and (2) that ND maintains some degree of affinity for each user. Web systems will perform better if most requests from a particular user go to the same IHS server. The ND sticky port option will be utilized. One should bear in mind that while this balances client IP addresses rather than balancing individual users it is sufficient for ITA purposes.
24	RAS – Process Failure: IHS	This test will verify that ND switches requests from an unreachable IHS server to a reachable IHS server when the IHS process fails.
25	RAS – Machine Failure: IHS	This test will verify that ND switches requests from an unreachable IHS server to a reachable IHS server when the IHS machine fails.
26	RAS – Network Failure: ND to IHS	This test will verify that ND switches requests from an unreachable IHS server to a reachable IHS server when the network connection between them is lost.
27	RAS – Machine Failure: ND	This test will verify ND takeover when one ND machine fails
28	RAS – Process Failure: ND	This test will verify ND takeover when ND processes on one machine fail.
29	RAS – WAS OSE Load Balancing to WAS from IHS	This test will verify (1) that WAS load balances (using the OSE plug-in) different user requests from the IHS servers across the available application server clones and (2) that the WAS OSE plug-in maintains some degree of affinity for each user. Web systems will perform better if most requests from a particular user go to the same application server. The WAS OSE session affinity option will be utilized. This option is on by default.

#	Scenario Name	Scenario Description
30	RAS – Process Failure: WAS Application Server	This test will verify (1) that the OSE can route traffic to a new application server when one WAS application server is no longer reachable and (2) that the WAS admin server automatically restarts the failed server within a reasonable period and (3) that the OSE resumes routing traffic to it. The application server must have multiple clones accessible to one IHS or this test is meaningless.
31	RAS – Process Failure: WAS Admin Server	This test will verify that (1) that WAS can continue to process requests in spite of a temporary failure of a WAS admin server and (2) that the WAS admin server is restarted automatically by the WAS nanny process within a reasonable period of time and (3) that once this restart occurs, WAS continues to function normally.
32	RAS – Machine Failure: WAS	This test will verify that ND can route traffic to a new IHS server when one WAS machine is no longer reachable. This scenario will require that ND is smart enough to determine that the IHS server is running, but that servlet requests to WAS are failing. Most likely, this will require custom advisors. If sufficient hardware resources are available, a single IHS machine may use more than one WAS machine. In that case, the WAS OSE plug-in rather than ND should reroute traffic.
33	RAS – Network Failure: IHS to WAS	This test will verify that ND can route traffic to a new IHS server when one WAS machine is no longer reachable. This scenario will require that ND is smart enough to determine that the IHS server is running, but that servlet requests to WAS are failing. Most likely, this will require custom advisors. Depending on the network topology, this test may be meaningless as it is quite likely that the other WAS machine is also unreachable
34	RAS – Process Failure: DB	This test will verify (1) that WAS can recover from an admin or session DB failure and (2) that the DB fail over procedures work. For simplicity, it will be assumed that the session DB and admin DB are on the same data server and thus fail and recover together. It is expected that when the DB ceases to function that WAS will soon also cease to function. How soon this occurs depends on the amount of caching and application usage patterns. In most cases, problems will begin to occur quickly. After the DB problem is corrected using the DB fail over procedures, WAS is expected to reconnect to the DB and then resume functioning normally after some period of instability.
35	RAS – Machine Failure: DB	This test will verify (1) that WAS can recover from a admin or session DB failure and (2) that the DB fail over procedures work. For simplicity, it will be assumed that the session DB and admin DB are on the same data server and thus fail and recover together. It is expected that when the DB ceases to function that WAS will soon also cease to function. How soon this occurs depends on the amount of caching and application usage patterns. In most cases, problems will begin to occur quickly. After the DB problem is corrected using the DB fail over procedures, WAS is expected to reconnect to the DB and then resume functioning normally after some period of instability.

#	Scenario Name	Scenario Description
36	RAS – Network Failure: WAS to DB	This test will verify that WAS can recover from a admin or session DB. For simplicity, it will be assumed that the session DB and admin DB are on the same data server and thus fail and recover together. It is expected that when the network connectivity to the DB ceases to function that WAS will soon also cease to function. How soon this occurs depends on the amount of caching and application usage patterns. In most cases, problems will begin to occur quickly. After the network problem, WAS is expected to reconnect to the DB and then resume functioning normally after some period of instability.
37	RAS – Disaster Recovery	This test will verify that the backup and recovery procedures actually work. This test should be performed on a regular basis to ensure that recovery procedures continue to work as the application environment changes. Define a working WAS environment with a few applications in it and then: 1. Perform a backup of all systems as defined in the backup procedures. 2. Forcibly remove the WAS admin databases using rm –r or equivalent. 3. Forcibly remove the WAS local configuration files on an application server machine. 4. Wait while everything goes to pieces. 5. Stop everything. 6. Execute the disaster recovery plan. This will include restoring system configuration from tape. 7. Verify that things now work.
38	RAS – Long Run Burn-in Test	This test is intended to generate a load that reflects the predicted usage of the actual production environment and thus should be executed in a test environment that is as similar as possible to the production environment. The system will be stressed such that CPU usage is at least 75%. This test will run continuously for 36 hours. During this time, it is expected that there may be a few errors. However, the total number of unexpected results should be very small. How small will vary depending on business requirements. This test will require the use of an automated load generation tool and a large number of test scripts that reflect the expected usage patterns.
39	RAS – Scalability	This test will verify that the WAS applications do scale. This relies on the work done previously to test load balancing. An application will be deployed with one clone on one machine in a 3 node WAS cluster. Using a tool like Load Runner, system performance (average latency, peak latency, through-put) will be collected. Find the sweet spot in the performance curve where the machine CPU is nearly saturated (90% +), but latency is still holding steady. Then, add a second application clone on the same node. Rerun the tests with increased load. Add clones to the second and third nodes and rerun the performance tests. A well-designed application should scale nearly linearly. Record these numbers for future reference in capacity planning.

5 Acronyms

Table 64 – Acrnoym List

Acronym	Description
AE	Advanced Edition
API	Application Programming Interface
BTR	Build and Test Report
CDS	Central Data System
CGI	Common Gateway Interface
CPU	Central Processing Unit
DB	Database
DRE	Data Reasoning Engine
ETL	Extract, Transform and Load
FTP	File Transfer Protocol
GUI	Graphical User Interface
HP	Hewlett-Packard
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IHS	IBM HTTP Server
IIS	Internet Information Server
IP	Internet Protocol
IT	Information Technology
ITA	Integrated Technical Architecture
JDBC	Java Database Connectivity
JSP	Java Server Page
LAN	Local Area Network
ND	Network Dispatcher
NT	New Technology
ODBC	Open Database Connectivity
OLAP	On-Line Analytical Processing

Acronym	Description
OSE	Open Servlet Engine
RAS	Reliability, Availability and Scalability
ROLAP	Relational On-Line Analytical Processing
SFA	Student Financial Assistance
SSI	Server Side Includes
TIVWAN	Title IV Wide Area Network
URL	Uniform Resource Locator
VDC	Virtual Data Center
VIC	Viador Information Center
WAN	Wide Area Network
WAS	WebSphere Application Server